UNIVERSITY OF MINNESOTA

PB99-163834

CENTER FOR TRANSPORTATION STUDIES

ITS INSTITUTE

ACTIVATION OF THE I-394 LABORATORY FOR ITS OPERATIONAL TESTING: PHASE 2

Kenneth Reynhout, Panos Michalopoulos, & Alexander Siagian

Department of Civil Engineering

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Research Institute Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

Technical Report Documentation Page

1. Report No.	2.	3. Recipient's Accession No.
CTS 98-02		

4. Title and Subtitle	5. Report Date
Activation of I-394 Laboratory for ITS Operational Testing (PhaseII)	April 1998
	6.

7. Author(s)	8. Performing Organization Report No.
Kenneth Reynhout, Panos Michalopoulos, and Alexander Siagian	

9. Performing Organization Name and Address	10. Project/Task/Work Unit No.
Department of Civil Engineering University of Minnesota 500 Pillsbury Dr SE Minneapolis, MN 55455	11. Contract (C) or Grant (G) No. (C) (G)

12. Sponsoring Organization Name and Address	13. Type of Report and Period Covered
Center for Transportation Studies ITS Institute Program 200 Transportation and Safety Building 511 Washington Avenue SE Minneapolis, MN 55455	Final Report
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract (Limit: 200 words)

The key element in improving traffic operations and performing real-time management is the ability to assess the effectiveness of various alternatives prior to implementation. Likewise, the crucial feature for providing this capability is a Traffic Data Management System (TDMS), which gathers data and makes it available for various traffic analysis applications. The purpose of this research is to develop TDMS as part of a Laboratory Environment for TRaffic ANalysis (LETRAN). Such a laboratory environment would provide easy and efficient access to various kinds of traffic data for use in simulation, control, incident detection, and other types of traffic analysis applications to be deployed in a next-generation traffic management center. In addition, a Machine-Vision Laboratory (MVL) will be designed and implemented as part of the Center for Transportation Studies(CTS) Intelligent Transportation Systems Laboratory (ITS Lab). This MVL will use live video feeds from both freeways and arterial streets and provide machine-vision technology for conducting traffic detection, data collection, and group training exercises. Such capabilities will allow for the collection of detailed, accurate, and continuous data for successful model development, calibration, testing and evaluation.

17. Document Analysis/Descriptors	18. Availability Statement
Traffic Management Center traffic-management transportation -planning	No restrictions. Document available from: National Technical Information Services, Springfield, Virginia 22161

19. Security Class (this report)	20. Security Class (this page)	21. No. of Pages	22. Price
Unclassified	Unclassified		

ACTIVATION OF THE I-394 LABORATORY FOR ITS OPERATIONAL TESTING

Final Report for Phase 2
Development of a Traffic Data Management System

Prepared by

Kenneth Reynhout, Panos Michalopoulos, and Alexander Siagian
Department of Civil Engineering
University of Minnesota
Minneapolis, MN 55455

July 1999

Submitted to

Minnesota Department of Transportation
Office of Research Services
395 John Ireland Blvd
St. Paul, MN 55155

Published by

Intelligent Transportation Systems Institute of the
Center for Transportation Studies
University of Minnesota
200 Transportation and Safety Building
511 Washington Avenue S.E.
Minneapolis, MN 55455-0375

ACKNOWLEDGEMENTS

This research was supported by the Minnesota Department of Transportation (Mn/DOT) and the Intelligent Transportation Systems (ITS) Institute in the Center for Transportation Studies (CTS) at the University of Minnesota. The authors would also like to thank the following individuals for their contributions to this document.

James Aswegan – *Minnesota Department of Transportation*

Lowell A. Benson – *Center for Transportation Studies*

Marcus J. Culver – *Image Sensing Systems, Inc.*

Ron Dahl – Mn/DOT's *Traffic Management Center*

John Hourdakis – *University of Minnesota Department of Civil Engineering*

Dr. Eil Kwon – *Center for Transportation Studies*

Dr. Shashi Shekhar – *University of Minnesota Department of Computer Science*

Dr. Jaideep Srivastava – *University of Minnesota Department of Computer Science*

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

EXECUTIVE SUMMARY

One of the most important aspects of advanced traffic engineering research is the ability to collect comprehensive, detailed, and accurate traffic data over large areas. This information is invaluable for researchers who are delving the depths of traffic flow theory, driver behavior, signal control, and traffic management. It is also crucial for real-time traffic management and off-line transportation planning. The University of Minnesota has therefore initiated a research program with the ultimate objective of investigating and developing new traffic data collection and management technologies to support decision making in a next-generation traffic management center. This work will have immediate benefits for research conducted as part of the ITS Institute in the Center for Transportation Studies at the University of Minnesota, and potentially long-term benefits as prototype systems for the next-generation traffic management center.

This project is the second of multiple research phases, and is therefore referred to as "Phase 2." The specific work required in Phase 2 included:

- To design and develop a Traffic Data Management System for loop-sensor data collection, management, distribution, and storage. The system should be easy to use, available to a general audience, and scaleable to a large operational system. Furthermore, the design of the system should incorporate relevant work from other research projects supported in the Center for Transportation Studies.
- To design and implement a Machine-Vision Laboratory that will use live video feeds (from I-394 or elsewhere) and local machine-vision devices to configure detection experiments and collect data for research. In addition, the lab will be useful for machine-vision training and instruction.

Future phases of this research will involve the integration of multiple types of traffic data and systems, such as geometry data, real-time sensor data, historical sensor data, control scenarios, microscopic simulation, and macroscopic simulation for use in real-time traffic management activities.

The project (Phase 2) was a success. The work completed and corresponding benefits include:

- The design of a Traffic Data Management System for loop-sensor traffic data, which integrates work from multiple database-related projects into a single system for collecting, distributing, and archiving loop-sensor traffic data.
- As a generous contribution to this design, the creation of a Real-time Data Distributor that provides straightforward access to live loop sensor data across various platforms, operating systems, and types of software applications. This revolutionary approach to live data distribution uses current standard protocols and object technology to produce an open distributed system. Such a system anticipates future growth, and provides a working a deployable information system for the next-generation traffic management center.
- The creation of a Machine-Vision Laboratory for education and research that uses live traffic video feeds and local machine-vision devices. Such a lab will enhance the learning and use of this advanced data sensing and collection technique.

1 Introduction

1.1 Research Objectives

This and other related research projects have two primary objectives:

1. To develop practical operational tools which can be deployed for use in traffic-management and transportation-planning activities, and
2. To develop a laboratory infrastructure which will facilitate future advances in traffic modeling and other ITS initiatives.

The general objective of the *I-394 Laboratory* research project is:

- To develop advanced traffic sensor data collection, management, and distribution systems that provide accurate, flexible, and wide-area traffic information.

The I-394 Laboratory research project was divided into project phases, outlined as follows:

Phase 1: I-394 Machine-Vision System analysis and I-394 Lab activation, as described in Section 1.2.

Phase 2: Traffic Data Management System design and development, as described in this report.

Phase 3: Integration of I-394 Lab with the results of other projects, including Simulation Lab and TRACLAB, as described in the conclusion of this report.

1.2 Background

The I-394 Machine-Vision System (hereafter referred to as the *I-394 MVS*), was designed around and built into the new I-394 infrastructure (completed in 1993). It was comprised

of 36 video cameras placed along over seven miles of the I-394 freeway (3.5 miles in each direction, and a separate High Occupancy Vehicle (HOV) lane), roughly between Penn and Louisiana Avenue. Nine machine-vision devices were deployed (six in the field and two at the Traffic Management Center (TMC)) to collect data from the 36 cameras, and transmit it in real-time to the TMC using fiber-optic lines. Unfortunately, for reasons that will not be discussed in this report, the I-394 MVS was never used effectively, and fell into disuse.

1.2.1 Phase I Objectives

The general objective of *Phase 1* of this research was:

- To develop a laboratory environment that uses machine-vision to collect and manage accurate, detailed, and continuous traffic data over long stretches of freeway for use in the design, development, calibration, testing, and evaluation of traffic models, traffic management systems, and other ITS applications.

The specific plan for meeting this objective was:

- To "resurrect" the dormant I-394 MVS and activate the "I-394 Lab" by establishing communications to and control over the MVS from the ITS Laboratory (ITS Lab). The I-394 Lab should be capable of controlling the I-394 MVS from the ITS Lab, including control over video, detector types/placement, and data collection instructions.

1.2.2 Phase 1 Results

Completing Phase 1 as planned proved to be much more difficult than expected. To begin with, the MVS was developed in two distinct stages, resulting in an ill-designed

and difficult-to-use system. To make matters worse, the system was poorly documented. By far, the biggest problem was that the MVS was never maintained, and was in dire need of repair. To help rectify these issues, the following work was completed:

- A thorough examination and documentation of the I-394 system's specifications, most of which did not exist prior to the project.
- Diagnostics of the working condition of the system (dormant for nearly three years). A number of problems were detected among cameras, communications, and machine-vision devices. It appears the communications problems were the most common and difficult to repair. Also, some of the computer equipment/software had become outdated, and needed upgrading.
- A repair plan was negotiated and attempted. Some of the aforementioned problems were fixed, including the machine-vision devices, some bad cabling, some upgrades to computers, and some system-design changes. Many of the problems, including the communications, were beyond the project's control and therefore not thoroughly rectified.
- Two communication connections were established between the ITS Lab and the TMC.
- A microwave system was used to transfer the live I-394 video to the ITS Lab.
- "Remote-control software" was used, which uses a regular telephone line and a modem to allow a user to control a Personal Computer (PC) from a remote location. In this way, control was gained over much of the I-394 machine-vision system. Completion of this task inaugurated the "I-394 Lab."

- The I-394 Lab was activated. A user's guide was written which describes the steps a user should take to access video, make a connection to the TMC and the I-394 system, and configure the system for traffic detection and data collection. Any data collected in this way will be stored in a supervisor computer at the TMC, and the user can use the remote control software to transfer the data files to the ITS Lab.

1.2.3 Phase 1 Conclusions

The aforementioned problems with the I-394 MVS led us to believe that the I-394 Lab could not be used effectively until improvements are made to the system, and funding for ongoing maintenance and support is in place. Nevertheless, smaller-scale experiments were still possible using the live video capabilities and local machine-vision devices in the ITS Lab. Phase 2 of this research followed through on that plan, and also expanded its focus to include general traffic-data management issues.

1.3 Phase 2 Objectives

The objectives of *Phase 2* of this research, the results of which are described in this report, were:

1. To design and develop a Traffic Data Management System for loop-sensor data collection, management, distribution, and storage, and
2. To develop a Machine-Vision Lab for training and research using the available I-394 camera feeds and local ITS Lab machine-vision processors;

And the specific tasks for this project, as outlined in the project plan, were:

1. Establish the requirements of the TDMS. (See Chapter 2)

2. Review ITS database projects and Mn/DOT systems for applicability to TDMS. (See Chapter 3)
3. Design TDMS. (See Chapter 4)
4. Implement TDMS in ITS Laboratory. (See Chapter 5)
5. Determine the requirements of the Machine-Vision Lab (MVL). (See Chapter 6)
6. Design the MVL. (See Chapter 7)
7. Implement the MVL. (See Chapter 7)
8. Design an access protocol that enables applications to communicate with the TDMS. (See Chapter 4)
9. Implement the TDMS access protocol. (See Chapter 5)
10. Identification and implementation of a GUI for TDMS. (See Section 4.7)

1.4 Report Organization

This report documents the final results from the I-394 Laboratory project (Phase 2) in the following order:

CHAPTER 1: This chapter – an introduction to the project.

CHAPTER 2: A description of the Traffic Data management System requirements.

CHAPTER 3: A review of related database project work.

CHAPTER 4: A description of the Traffic Data management System design specifications.

CHAPTER 5: A description of the Traffic Data management System implementation, including

CHAPTER 6: A description of the Machine-Vision Lab requirements.

CHAPTER 7: A description of the Machine-Vision Lab design specifications and implementation.

CHAPTER 8: The conclusion of the report, including recommendations and a discussion of future direction.

APPENDIX A: A discussion of ITS deployment standards that had bearing for this project, including the National ITS Architecture and the National Transportation Communications for ITS Protocol (NTCIP).

APPENDIX B: A discussion of CORBA and software integration, and the subsequent impact on ITS development.

APPENDIX C: A list of stations and detectors, describing their locations and relationships to one another.

APPENDIX D: The source code for the 30-second RDD server application.

APPENDIX E: The source code for an example 30-second RDD client application.

APPENDIX F: Details regarding the use of the Autoscope ScopeServer aplications.

2 TDMS Requirements

2.1 Data Types and Storage

The Twin Cities' freeway loop detector data comes in two formats: five-minute detector data and thirty-second station data.

2.1.1 Five-minute Data

The five-minute data comes from 3500 individual detectors, typically placed in each lane of the freeway every half-mile and on all exit and entrance ramps. (See Appendix C for a description of the detectors and locations.) The detector data is collected every five minutes, and consists of a volume, occupancy, and two flags. The sizes, ranges, and meanings of these different values are as follows:

Volume	8 bit unsigned integer	0 to 255	Number of vehicles that passed over the detector in the last five minutes
Occupancy	7 bit unsigned integer	0 to 100	Percentage of time that the detector has been occupied
Status	3 bit signed integer	-4 to 3	Communication status
Validity	3 bit signed integer	-4 to 3	Data validity

Table 2.1 – Five-minute data values

The status flags can take the following possible meanings:

Status	-4	Checksum
	-3	Timed out
	-2	Line fail
	-1	Fail
	0	No error
	1	Controller flagged the data
	2	Future
	3	Undefined detector

Table 2.2 - Status flag values

If the five-minute data were collected and stored in its smallest possible format (as binary data), every individual five-minute detector data packet would require 21 (8+7+3+3) bits, or 2.625 bytes, of storage. For all 3500 detectors, 3500*2.625 = 9,187.5 bytes of storage space required for every five minutes. Over time, this translates to roughly 966 megabytes of storage per year.

5 minutes	1 hour	1 day	1 week	1 month	1 year
0.0092 MB	0.11 MB	2.65 MB	18.5 MB	79.4 MB	965.8 MB

Table 2.3 – Five-minute storage requirements

2.1.2 Thirty-second Data

The thirty-second data comes from 864 detector stations, and is collected every thirty seconds. Stations are used to aggregate detector data across lanes, at a particular directional location along a freeway. Figure 2.1 illustrates a station averaging data across three lanes.

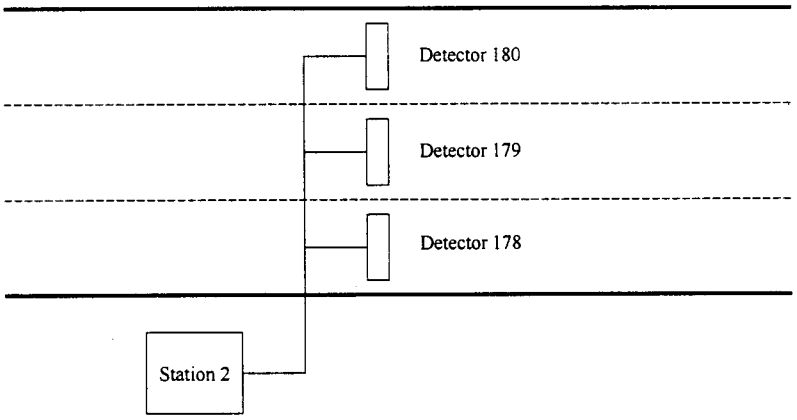


Figure 2.1 – Detector/Station Relationship

The collected data consists of a thirty-second volume, occupancy, and a status flag. The sizes, ranges, and meanings of these different pieces of information are as follows:

Volume	6 bit unsigned integer	0 to 63	Average of the individual detector volumes
Occupancy	7 bit unsigned integer	0 to 100	Average of the individual detector occupancies
Status	3 bit signed integer	-4 to 3	Communication or detector status

Table 2.4 – Thirty-second data values

See Table 2.2 for status flag meanings.

If the thirty-second data were collected and stored in its smallest possible format (as binary data), every individual thirty-second station data packet would require 16 (7+6+3) bits, or 2 bytes, of storage. For all 864 stations, 864*2 = 1,728 bytes of storage space required for every thirty seconds. Over time, this translates to roughly 1.8 gigabytes of storage per year.

5 minutes	1 hour	1 day	1 week	1 month	1 year
0.017 MB	0.21 MB	4.8 MB	34.8 MB	149.3 MB	1,816.5 MB

Table 2.5 – Thirty-second storage requirements

Adding together five-minute and thirty-second data, approximately 2.8 gigabytes of storage would be required per year.

5 minutes	1 hour	1 day	1 week	1 month	1 year
0.026 MB	0.32 MB	7.6 MB	53.4 MB	228.7 MB	2,782.3 MB

Table 2.6 - Total storage requirements

In all likelihood, the five-minute and thirty-second data would not be stored in binary format, but rather in tables as part of a relational database. This format would increase the storage requirements just outlined.

2.2 *Time Dependent Utilization*

The utilization of loop detector data changes over time, and different ITS operational tools and research methodologies will have different data age-utility requirements. There are three general data categories that help to distinguish among these different age-utility requirements: real-time, active, and archived data.

2.2.1 Real-time Data

Real-time Data – data to be used as soon as its available. Some applications that may need real-time data are:

- Incident detection
- Ramp metering
- Traffic monitoring
- Variable message sign control

2.2.2 Active Data

Active data is no longer real-time, but is "young" enough to make it potentially valuable. The age at which data is no longer considered to be young should be determined by the system capabilities and user needs (e.g. one month, one year). Some applications that may need active data are:

- Simulation/control systems
- Model testing and calibration
- Congestion pricing tests
- Regular report generation

2.2.3 Archived Data

Archived data is no longer "young", and is stored for the long-term. Its existence and availability are more important than easy accessibility. Some applications that may need archived data are:

- Testing traffic variations due to weather
- Traffic growth pattern research
- Seasonal demand/capacity variation
- Road/freeway design
- Land use change impact

Some applications may require more than one or all three of these data types.

2.3 Usability

The TDMS should be straightforward and intuitive, easy to use for an average user (someone without programming expertise, for example). Some of the ways this could be achieved include:

- The use of graphical and map-based interfaces
- The use of object-oriented techniques and interfaces
- Transparent access to data, by hiding details from users
- The use of programming techniques which hide client-server networking details from users

2.4 Deployment

2.4.1 ITS Laboratory

Deployment of the TDMS in the ITS Lab was complicated by the distributed nature of the laboratory. The ITS Lab has over fifteen networked computers of various types and operating systems, including UNIX (SUN, HP, and SGI) and personal computers (Windows 3.1, 95, and NT). (See Figure 2.2)

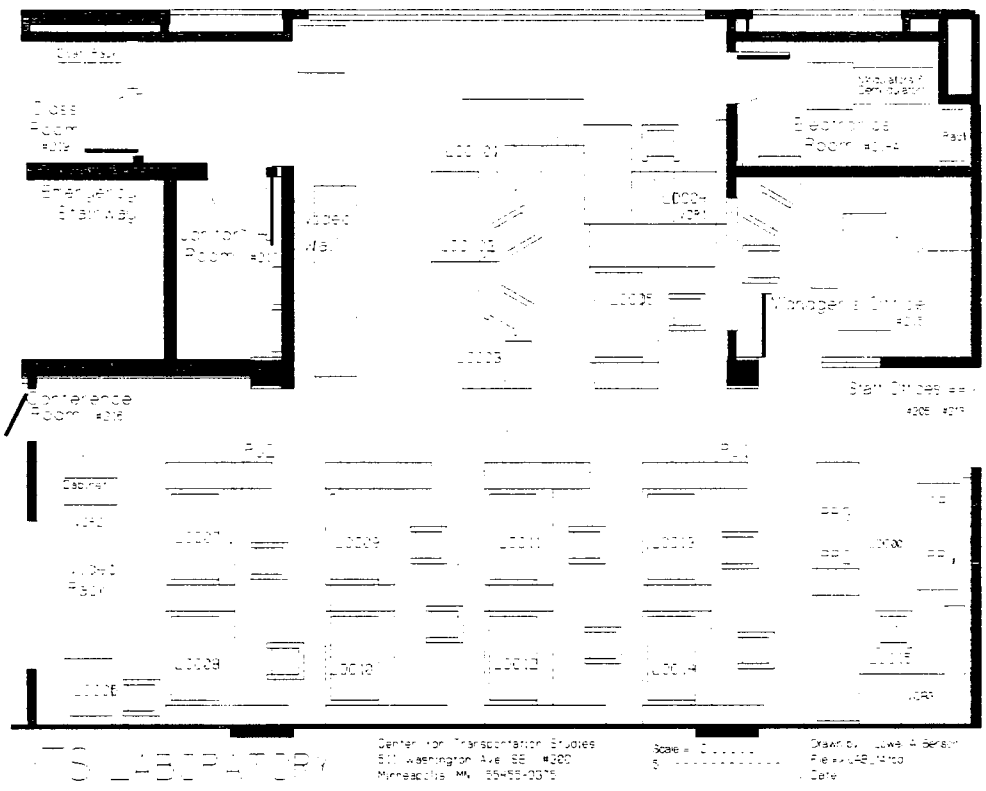


Figure 2.2 – ITS Laboratory Layout

The lab services a number of constituents with differing backgrounds and needs: professors, researchers, students, visiting scholars, and government employees, coming from transportation, engineering, computer science, human factors, and other fields. Furthermore, each constituent may have unique software needs (programming languages,

operating systems, etc.) and differing levels of sophistication. If possible, the TDMS design should also anticipate possible future expansion of the ITS lab usage or growth in scope.

2.4.2 Mn/DOT and TMC

The TDMS should be deployable as an operational management tool for Mn/DOT, the TMC, and other future management center designs. Mn/Dot and the TMC use PC-based computer systems, and an Oracle database environment. The TDMS should use standards that will conform to these systems, but will remain flexible enough to be used in the next-generation traffic management center. In short, we were faced with the task of designing a straightforward, integrated, open, flexible, and scaleable information system for traffic data management, similar to what would be encountered in designing, installing, and maintaining the next-generation traffic management center.

2.5 Integration of Existing Projects/Work

The design of the TDMS should also integrate existing or ongoing work from other CTS-related database projects and initiatives. The following chapter will review these relevant projects.

3 Database Project Review

3.1 Data Distribution Server

The Mn/DOT Traffic Management Center's Data Distribution Server (DDS) provides access to data from TMC systems, including five-minute detector, thirty-second station, and ramp-metering rate data. (A future version of this system will also broadcast incident data.) The DDS uses the TCP/IP communications protocol to send information to interested clients. Clients connect to the DDS using a named pipe and sockets programming. Information is broadcast to clients by the DDS as a stream of continuous information, and clients break this stream apart into its individual pieces. (See Figure 3.1)

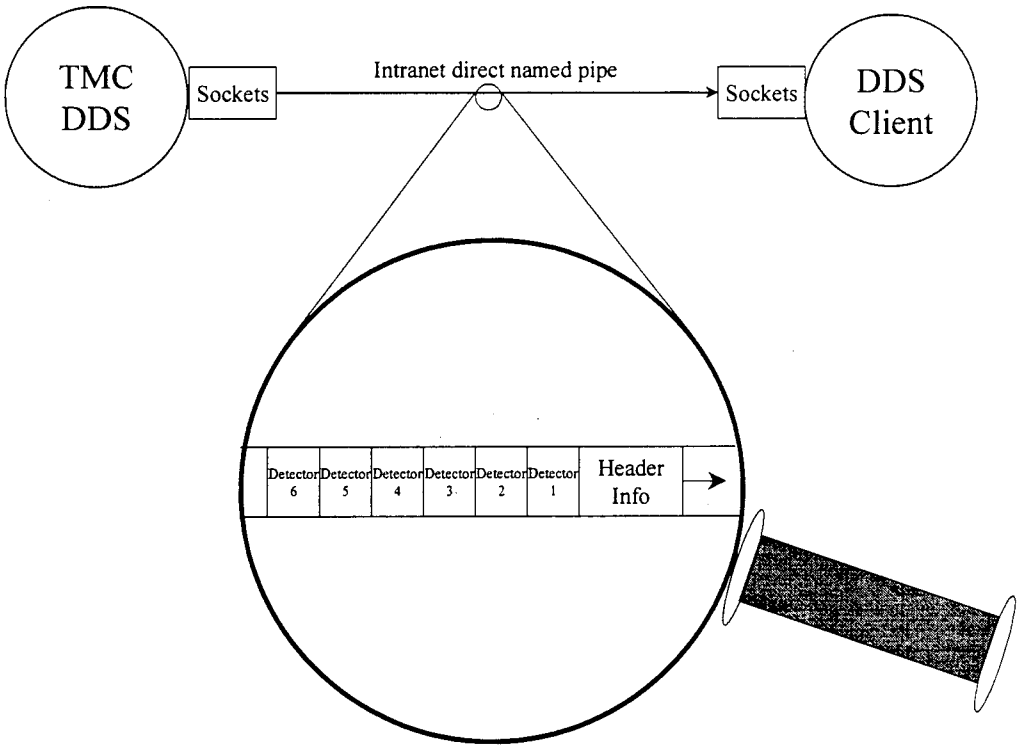


Figure 3.1 – The DDS System

Although the sockets approach for broadcasting data is effective for long-distance internet communications, it has limitations as a real-time broadcast methodology in either a laboratory or management center environment. Specifically:

- Sockets are difficult to understand and program with, which is a serious issue for engineers who are not necessarily expert programmers
- Sockets are operating system dependent (sockets programming in Windows NT is not the same as sockets in UNIX)
- Only one kind of connection (detector, station, or ramp meter) can be made at a time
- *All* of the data is sent to clients, even if only a subset is desired, which could overload a local area network (e.g. all 3500 detector data packets are sent every five minutes)
- There are a limited number of connections that can be maintained simultaneously

3.2 Traffic Data Archival Project

In 1995, a cooperative effort between the TMC, Mn/DOT's Freeway Operations group, and the University of Minnesota's Professor Shashi Shekhar (Computer Science) was begun to develop a database system that archives loop detector data. The resulting system (completed in late 1996) used a PC-based ORACLE relational database environment to store and manage five-minute detector data. The database was designed to not only archive the raw detector data; a lot of summary information was created and stored as well. The summary information was intended to assist Mn/DOT in analyzing and reporting the data, and usually centered around summarizing measurements for different detector groupings (stations), planning zones, operational zones, and common

traffic routes for different time periods (15 minute, 1 hour, 1 day, etc.). The resulting database grew quite large – approximately 40 megabytes per day, or 15.6 gigabytes per year, of only five-minute data! Consequently, querying the database became a time-expensive task, and the performance suffered. Mn/DOT has since negotiated with Prof. Shekhar to help them redesign and re-implement the archival database to a higher performance level. The resulting system will reside in the CTS ITS Laboratory.

3.3 DATATOOL

The Data Retrieval and Analysis Tool (DATATOOL) is an application developed by Professors Slagle and Srivastava in the Computer Science Department at the University of Minnesota. DATATOOL combines two other systems developed at the University: the Data Flow Query Language (DFQL) and the Myriad Federated Database System (Myriad). DFQL is a user-friendly graphically based method for querying databases and viewing the results. It replaces SQL for the average user. Results can be viewed in two- and three-dimensional graphs. Myriad is a tool for creating a federated database, a "meta-database" comprised of multiple smaller databases. Myriad is useful because it hides the complexity of using multiple data sources from the user, so that all the user sees is one large database. DATATOLL, combining these two powerful tools, should prove very useful in the overall design of the TDMS, which will become clear in the following chapter.

4 TDMS Design

The design of the TDMS focused on three themes: 1) integrating existing relevant work; 2) keeping the system open and flexible by relying on distributed client-server computing techniques; and 3) enabling integration and scalability by using standards wherever possible. The data-flow of the developed design is summarized in Figure 4.1. Different parts of this design will be discussed individually.

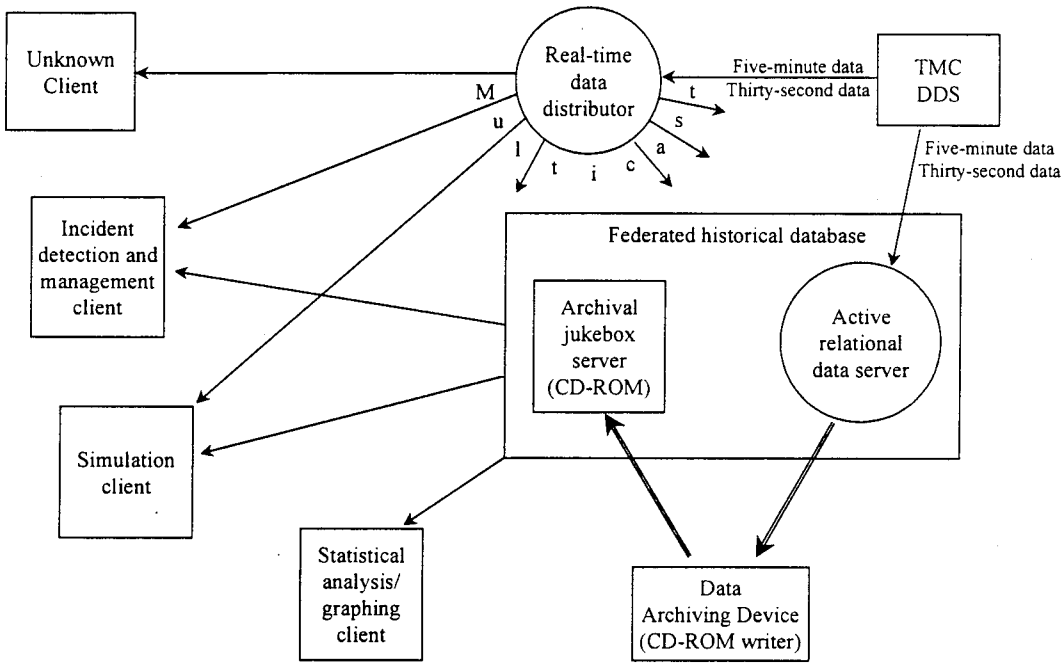


Figure 4.1 – TDMS Data-flow Design

4.1 TMC DDS Live Data Feed

Live five-minute and thirty-second data will be received from the TMC via the DDS described earlier. Two DDS client programs will have to be written which use sockets to

extract the data from the intranet connection, one for the Real-time Data Distributor (RDD) and the other for the federated historical database. It was decided to have two direct DDS feeds since both clients need *all* of the data for their processing and storage activities. Note that this design only accommodates loop sensor data, but it is open to adding other types of data, such as ramp-metering or incident data.

4.2 Real-time Data Distributor

The Real-time Data Distributor (RDD) is responsible for collecting information from the TMC DDS (via sockets), and then disseminating this information to interested users in real-time. This system should be a *multicast* system. A multicast system delivers information only to users who are interested in it, instead of a *broadcast* system that delivers to all of the users. In addition, this system should follow a *publish-subscribe* model, which allows users to subscribe for and receive pieces of the data instead of receiving *all* of the data. Finally, this system should work in real-time, and be *event-driven*. In other words, the *event* characterized by the arrival of data from the TMC DDS should initiate a delivery to interested users. Users should only have to request data once, not every time they need it (like every thirty seconds or five minutes). Designing and developing this system became the primary work of this research, and is described in detail in chapter 5.

4.3 Active Data Server

The active historical database server will store and manage "recent" information, from the last month or more depending on the size and speed of the server chosen for implementation. It is assumed that this recent information is considered to be more

valuable for research and operations activities, and the emphasis is therefore put on fast data accessibility. Relational databases are widely accepted as the best database solution for managing temporal data of this sort, and it is recommended that an industry-standard implementation be chosen (such as Oracle, Informix, or Sybase).

Fortunately, the work being done by Professor Shashi Shekhar (Traffic Data Archival Project) to develop a loop-detector database system using PC-based ORACLE should be sufficient for the Active Data Server. This project is currently in progress. (See Section 3.2)

4.4 Archived Data Server

After "expiring" from the historical database, information should be saved on some sort of archival medium. We recommend compact discs (using a writeable CD_ROM device). Compact disks are a good choice because:

- They hold a large amount of information in a relatively small space (640 MB)
- They have a ubiquitous presence in the technology community
- DVD-ROMs will soon overtake CD-ROM technology, and using CD-ROM now has the best chance of being upgradeable to DVD-ROM (DVD disks hold 4 GB or more!)
- Jukebox servers exist for cataloguing and managing information on CD-ROM

The archived historical database server will therefore be a jukebox server with an appropriate front-end tool for querying information from this server. Implementation of this archival system is not as crucial as other components of the TDMS, and will not be attempted in this project.

4.5 Federated Database

The federated database should give clients transparent access to data from the active and archived data servers. For example, a query could be formulated to compare yesterday's traffic flows with last year's flows on the same day, and the TDMS would retrieve the information from the two sources without the user's explicit knowledge. This hides the complexity of the TDMS design from the average user, such that it appears to behave as a single database, even though it exists as two separate servers.

It is possible that this federated capability could be achieved by using the Myriad tool developed by Professors Slagle and Srivastava (see Section 3.3). They have a project which is attempting to install Myriad (through DATATOOL) in the ITS Lab. The creation of the federated database should be a goal of that research.

4.6 Implementation Standards

Two standards are recommended for implementation: TCP/IP for a network protocol (supported by NTCIP) and Open DataBase Connectivity (ODBC) SQL-based querying for relational database communications.

4.7 Data Clients

There are a number of possible clients who could be using the TDMS system, including,

- Simulation programs
- Incident detection/management systems
- Statistical analysis and graphing software

Concerning the statistical analysis and graphing client, the DATATOOL product created and maintained by Professors Slagle and Srivastava is a possible candidate (see Section

3.3). In particular, DATATOOL's unique graphical querying technique (using DFQL) could prove very powerful for average users who are not necessarily intimate with SQL. Recognizing this, it is the authors' recommendation that DATATOOL be utilized as the graphical user interface for the TDMS, and should be part of Professor Slagle's and Srivastava's current CTS project.

4.8 Project Emphasis

As previously mentioned, this research will focus on the design and development of the Real-time Data Distributor, which is described in detail in the next chapter. Other parts of the TDMS design were reserved for development by other projects and entities. The scope of work done in developing the RDD is much greater than originally proposed to accommodate the fact that some of the other proposed tasks are now being covered by other projects (such as the GUI).

5 TDMS Implementation

5.1 Real-time Data Distributor Requirements

The Real-time Data Distributor (RDD) needed to meet some requirements for installation in the ITS lab, and for potential future implementation in a management center. Specifically,

1. It must work in a distributed network of machines.
1. Each machine may run a different operating system, such as UNIX or Windows NT/95.
2. Each machine may have different software programs written in different languages, such as C, C++, Java, or Visual Basic.
3. Each machine may have different ITS systems needing real-time data, such as incident detection/management, simulation/control, or traffic monitoring systems.
4. Each ITS system should have the capability of requesting any subset of the real-time data. For example, a user may like to obtain real-time data from loop detectors along a freeway corridor for testing incident management/control strategies.
5. Programming with such a system should be relatively easy and straightforward (since traffic engineers may not be accomplished programmers).
6. The RDD must be scaleable - designed for future growth and integration of other systems.
7. The RDD should use standard protocols supported by the transportation industry. (See Appendix A for a discussion of transportation standards.)

These requirements are actually a microcosm of a general set of requirements for all integrated software systems. Appendix B describes this in more detail.

5.2 RDD Design

The RDD followed a *publish-subscribe* model. In other words, the RDD works as a real-time data broker between the TMC DDS sockets data stream and the lab users. Each user subscribes for the data pieces he/she is interested in, and the RDD will publish the data when it becomes available (see).

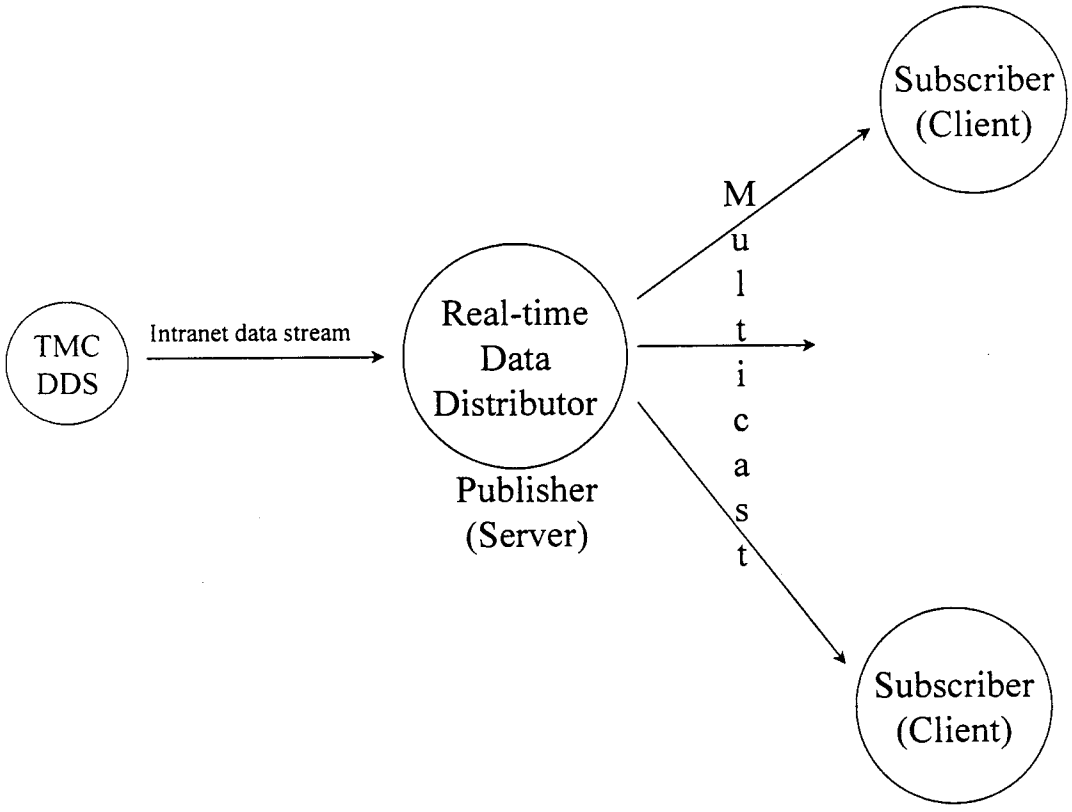


Figure 5.1 – Publish-Subscribe Model

In order to adhere to the publish-subscribe broker model, we decided to use the *Common Object Request Broker Architecture (CORBA)*. CORBA, described in detail in Appendix

B, is a set of standards developed in the early 1990s for distributed software systems. Some of the desirable attributes of CORBA are:

- CORBA is designed to work in a distributed network of machines.
- CORBA is operating system independent (unlike sockets).
- Implementations of CORBA operate with different programming languages, such as C++ and Java.
- CORBA is object-oriented, a programming methodology which has proven to be both powerful and intuitive.
- Implementations of CORBA simplify programming by hiding networking details from the programmer (unlike sockets and other inter-networking programming solutions).
- CORBA is being adopted as a standard software communications protocol as part of the National Transportation Communications for ITS Protocol (NTCIP). (See Appendix A for a more detailed description of NTCIP.)

Since CORBA is a standard protocol and not a programming language or software system, there are actually many proprietary implementations of the CORBA standard. These implementations are known as Object Request Brokers (ORBs). (See Appendix B for more a detailed description of ORBs.) This project chose to use the Orbix ORB, from Iona Technologies, Inc. Orbix is one of the most successful CORBA implementations, and enjoys a large market share.

CORBA also has useful extensions that cover common distributed software system scenarios. These extensions, the *CORBAservices* and *CORBAfacilities*, will not be discussed individually here, except for a particular service that was applicable to the

RDD: the *Event Service*. The Event Service facilitates the sending of messages to any number of receivers. Specifically, it allows software objects to dynamically register/unregister interest in specific events. An *event* is an occurrence specified to be of interest to one or more software components, such as the arrival of data. A *notification* is a message an object sends to interested parties informing them that a specific event has occurred. The service does this by defining two roles for software objects: *suppliers* and *consumers* (analogous to publishers and subscribers).

Figure 5.2 illustrates how CORBA and the CORBA Event Service were used to implement the publish-subscribe (supplier-consumer) model for the RDD.

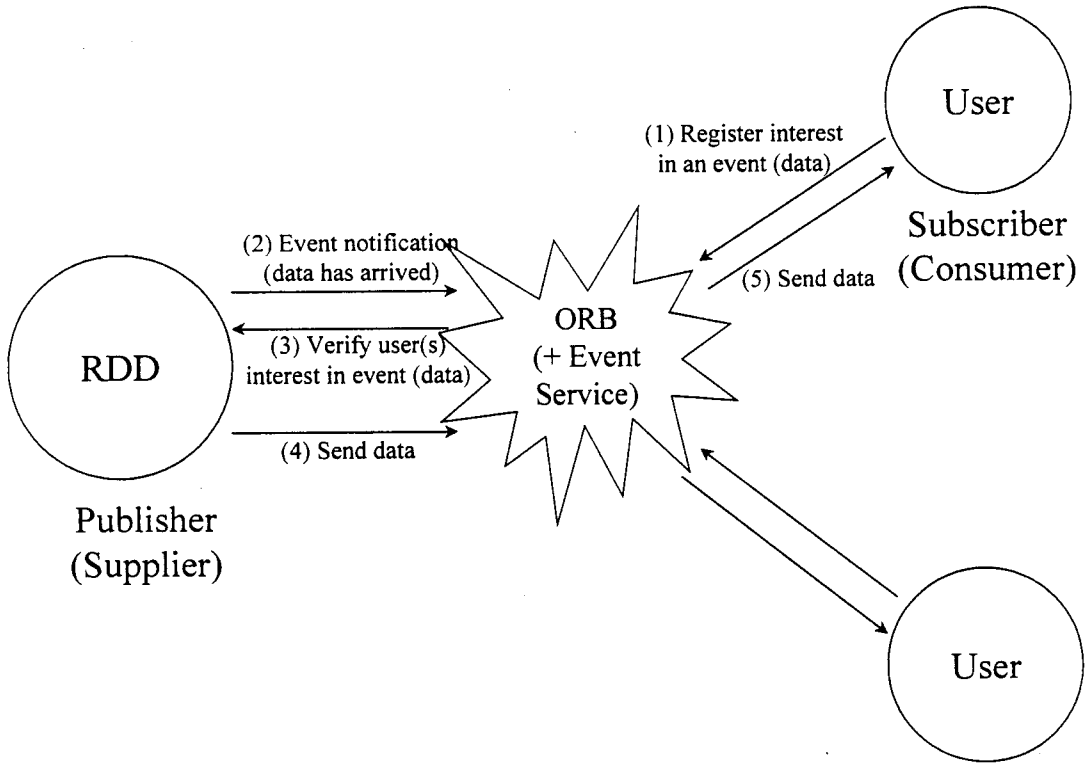


Figure 5.2 – RDD and the CORBA Event Service

Fortunately, Orbix has a proprietary implementation of the CORBA Event Service called *OrbixTalk*. Using OrbixTalk greatly simplified programming with the RDD.

5.3 RDD Implementation

Implementation of the RDD design involved two related tasks:

1. Writing the RDD Server program, which multicasts five-minute and thirty-second data.
2. Writing an example RDD client, which demonstrates how to program easily using the RDD.

5.3.1 The RDD Server

The RDD Server is responsible for:

1. Collecting the real-time data (using the TMC DDS data stream).
2. Packaging the data into individual pieces (determined by the detector or station id numbers).
3. Delivering the pieces to interested users.

The RDD Server accomplishes these tasks by using the sockets, the Orbix ORB, and the OrbixTalk event service. Specifically:

1. The Server begins by first connecting to the Orbix ORB, and then puts itself into an infinite waiting state. (This process must be spawned as a unique thread, or it will block other programs and disturb Windows event processing. See Appendix D for details.)
2. The Server then registers the OrbixTalk *names* of the data objects to be multicasted. (OrbixTalk names are described further below.)

3. The Server then makes the sockets connection to the TMC DDS, and waits for data to arrive.
4. When the data (five-minute or thirty-second) arrives, the Server then sends the data to any/all interested users. (This is accomplished by invoking the *deliver* function, which resides remotely in client applications.)
5. The Server then waits for the next data stream to arrive.

NOTE: Because only one DDS connection is allowed per application, there are actually two RDD Server programs running simultaneously, one for five-minute detector data and the other for thirty-second station data. From the client's perspective, however, it appears only one Server application is running.

5.3.2 RDD Data Objects

The data objects have been named in an attempt to both uniquely identify each data piece, and to also anticipate expansion into other real-time data types (such as different time intervals, sensor types, ramp-metering, or incident data). The names were chosen according to the following schemes:

- *loop\detector\5minute\id#*, where # can take any value 1-3500
- *loop\station\30second\id#*, where # can take any value 1-864

In addition, two other data objects exist for users who may want to subscribe for the entire set of a particular type of real-time data:

- *loop\detector\5minute\ALL*
- *loop\station\30second\ALL*

5.3.3 Running the RDD Server

To run the RDD Server, follow these steps:

1. Start the Orbix Daemon process, which can be found under Start => Programs => Orbix => Orbix Daemon. (See Figure 5.3.) Minimize the application to avoid cluttering the desktop.

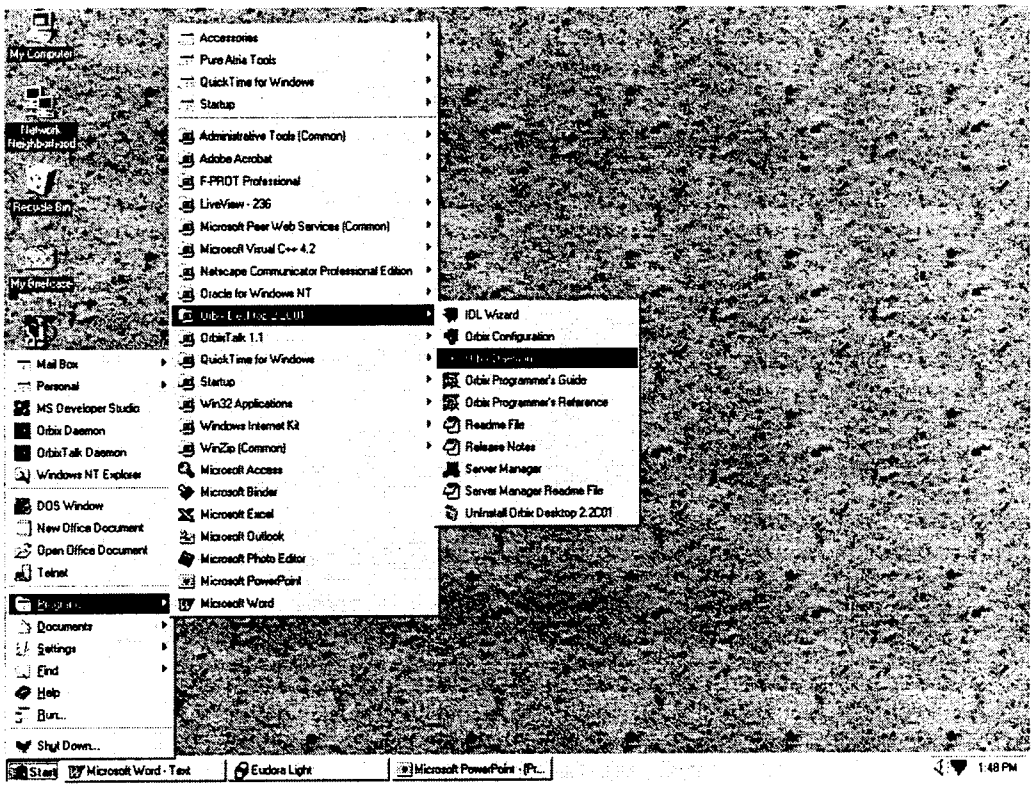


Figure 5.3 – Starting the Orbix Daemon

2. Start the OrbixTalk Daemon process, which can be found under Start => Programs => OrbixTalk = OrbixTalk Daemon. Wait a few minutes to allow the application to initialize and start the event service. You will know this has happened when messages appear in the DOS window indicating the both TopicStore and AppliStore have been opened. (See Figure 5.4.)

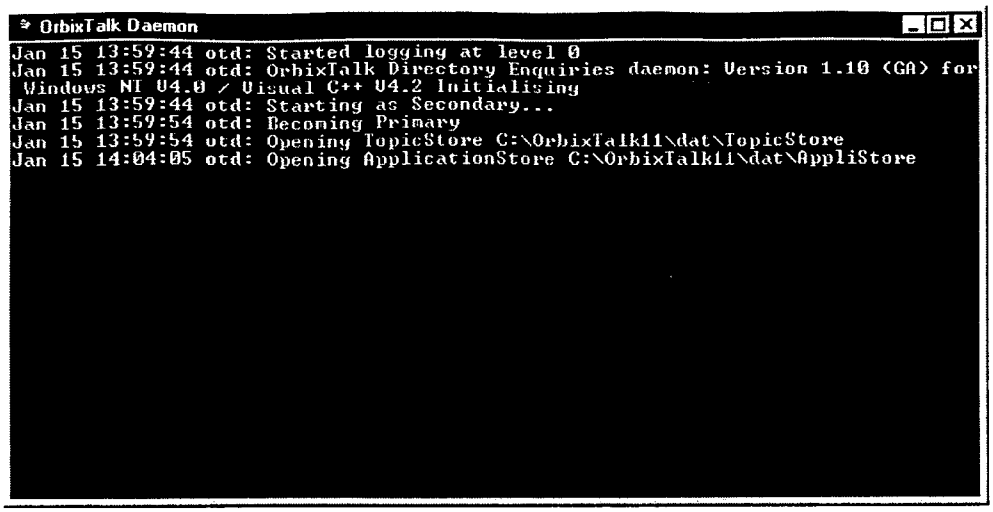


Figure 5.4 – Starting the OrbixTalk Daemon

3. Start one of the Server applications. (See Figure 5.5.)

   On the MARS workstation, these server programs can be found in:

   Five-minute: C:\msdev\projects\30secLoopServer\Release\LoopServer.exe

   Thirty-second: C:\msdev\projects\5minLoopServer\Release\LoopServer.exe

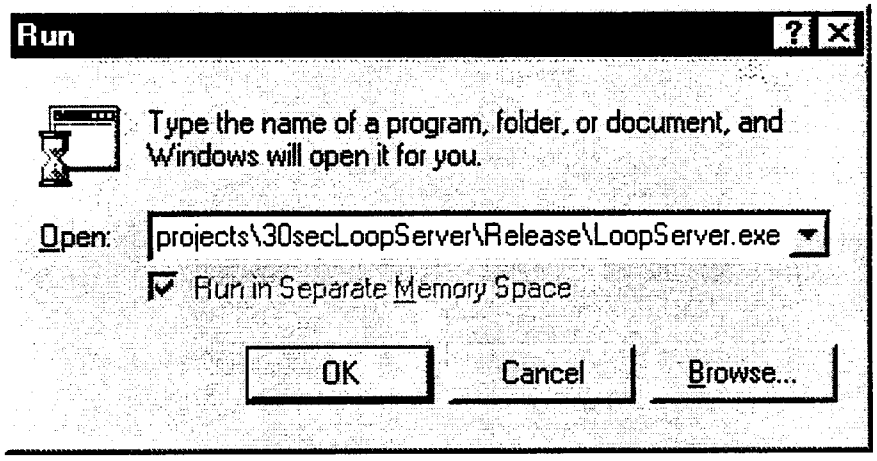


Figure 5.5 – Starting the Server Applications

4. Launching the multicasting requires two steps. First, under the File menu, choose the Register option. (See Figure 5.6.) If you look at the OrbixTalk Daemon at this time you will see it registering the OrbixTalk names. (See Figure 5.7.) There are many names to register (3500 for five-minute data and 864 for thirty-second data), and the registration process can take much time. This is not a major concern, however, because registration happens only once when the Server is started. Theoretically, you should never have to do this again.

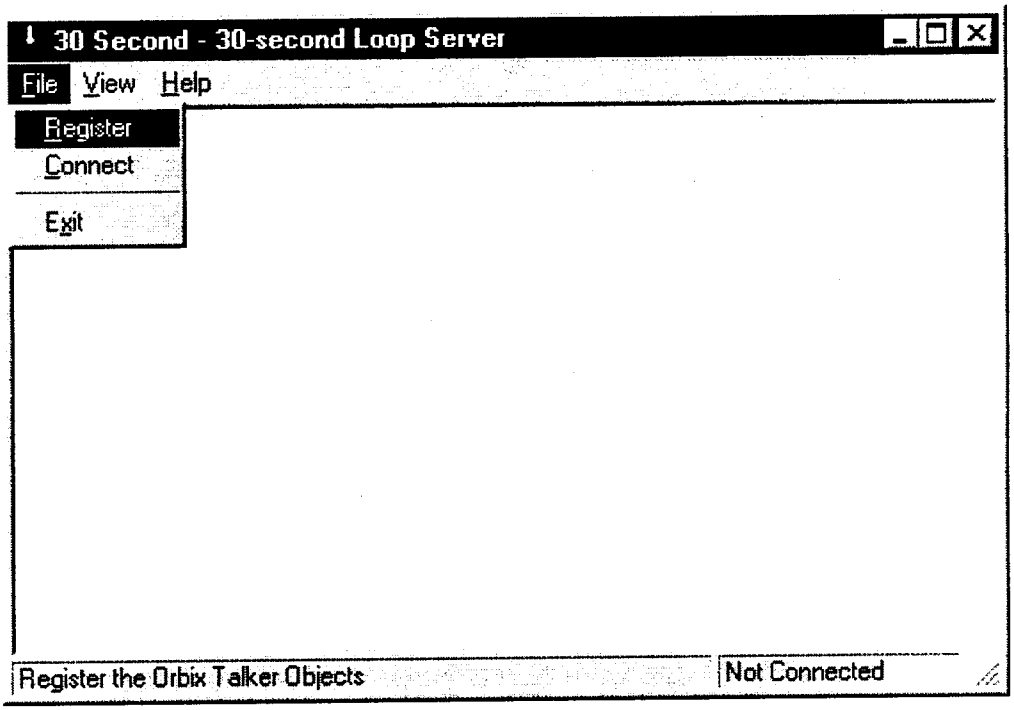


Figure 5.6 – Register the OrbixTalk Names

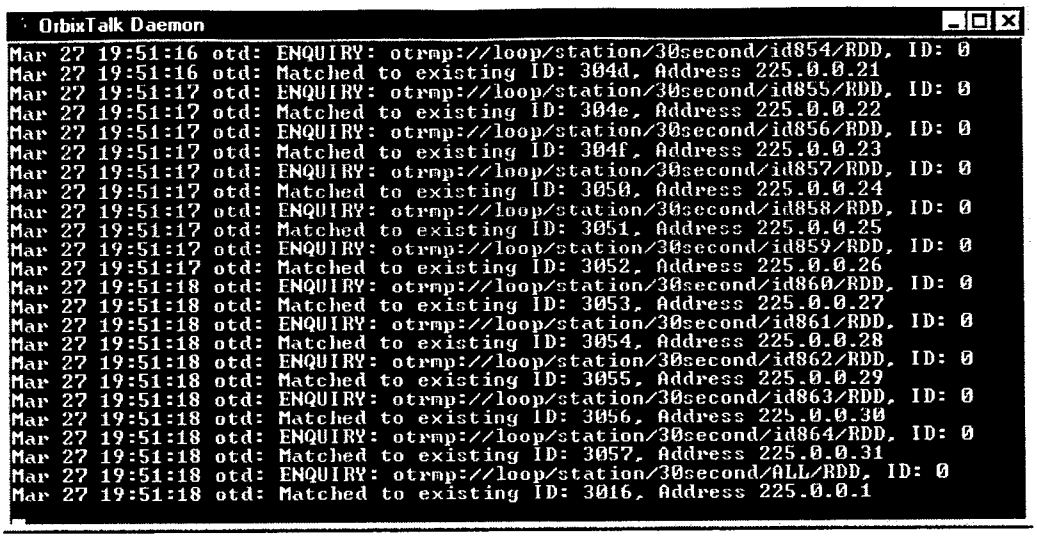


Figure 5.7 – Registering the OrbixTalk Names

5. The second step is to connect to the TMC DDS. Do this by choosing the Connect option under the File menu. (See Figure 5.8.) If the DDS is running properly at the TMC, the Server should now be running.
6. Now start the other Server application (Five-minute or Thirty-second) the same way as the first. When finished, minimize both Server applications and the OrbixTalk Daemon.

NOTE: Ideally, a Server application would be launched automatically when a network system is started. We did not attempt to create an automatic Server application in this project, but that could be accomplished with some effort.

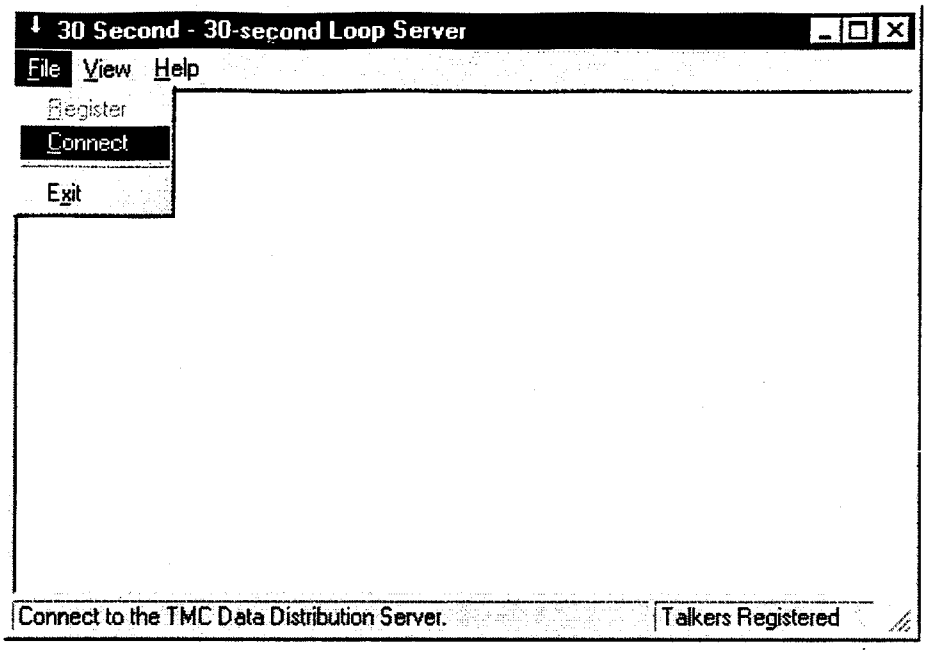


Figure 5.8 – Connecting to the TMC DDS

5.3.4 The RDD Client Sample

A typical RDD client application will use the Orbix ORB and the OrbixTalk event service to collect data from the RDD Server. Specifically:

1. A client begins by first connecting to the Orbix ORB, and then puts itself into an infinite waiting state. (This process must be spawned as a unique thread, or it will block other programs and disturb Windows event processing. See Appendix E for details.)
2. A client then registers its interest in particular OrbixTalk names.
3. When the data (five-minute or thirty-second) is available, the RDD Server sends it to the clients who have registered interest in particular data object names. (This is accomplished by invoking the *deliver* function, which exists in the client

application. The client programmer is responsible for adding appropriate code to this function in order to process the data when it arrives.)

5.3.5 Running the RDD Client Sample

The RDD system is most useful when employed "inside" of dynamic ITS applications by using the OrbixTalk programming methodology just described. Nevertheless, a client application sample was developed to illustrate a client program using the real-time data (thirty-second). Appendix E contains source code for the LoopUser client program. An experienced programmer will note the relative ease in programming a client application using this method.

To run the RDD sample client, follow these steps:

1. Start the Orbix Daemon process, which can be found under Start => Programs => Orbix => Orbix Daemon. Minimize the application to avoid cluttering the desktop. (See Figure 5.3.)
2. Start the OrbixTalk Daemon process, which can be found under Start => Programs => OrbixTalk = OrbixTalk Daemon. Wait a few minutes to allow the application to initialize and start the event service. You will know this has happened when messages appear in the DOS window indicating the both TopicStore and AppliStore have been opened. (See Figure 5.4.)
3. Start the 30-second Server application as described earlier. Register the topic names and connect to the TMC DDS.
4. Start the client sample application.

   On the MARS workstation, this sample client can be found in:

   C:\msdev\projects\LoopUser\Release\LoopUser.exe

The sample client application displays 30-second data for the first 100 stations. Note: Iona distributes its Orbix products in two forms: as a development license and as a run-time license. The development license is necessary when writing a program that utilizes the Orbix product, such as the aforementioned client. The ITS Lab currently has two of these licenses to pass around. To run a program that uses Orbix, a run-time license is required. This license is very inexpensive and cost-effective for an organization, such as a traffic management center, that might have a large number of such programs running.

6 Machine-Vision Lab Requirements

6.1 Functional Requirements

The Machine-Vision Lab (MVL) will have two primary purposes: research and education.

6.1.1 Research

One of the most important aspects of advanced traffic engineering research is the ability to collect comprehensive, detailed, and accurate traffic data over large areas. This information is invaluable for researchers who are delving the depths of traffic flow theory, driver behavior, signal control, and traffic management.

Historically, inductive loop detectors have been widely used as a traffic data collection technique. Loop detectors are physically installed in the pavement and detect vehicles that pass through their "detection zone" as defined by the magnetic field the loop generates. They are best utilized as vehicle counters, and can provide very useful information if placed across all lanes of the road.

Although loop detectors can be used to estimate speed and occupancy values, these estimates are not nearly accurate enough for advanced and precise research.

Furthermore, loop detectors cannot be used for wide-area detection. Loop detectors are difficult and expensive to install and repair, so using a large number of them over a wide area is not economically feasible.

Machine-vision offers researchers a chance to improve on some of the shortcomings of loop detection. Machine-vision uses video cameras, which can be placed at tightly spaced intervals along a roadway without tearing up pavement and interrupting traffic

flow. In addition, machine-vision technology can provide more accurate and comprehensive traffic data in very short time intervals, including:

- vehicle presence and passage
- vehicle speed and length classifications
- traffic flow rate, volume, lane occupancy
- traffic headway over time and level of service

One primary requirement of the Machine-Vision lab is to provide researchers with the opportunity to collect and use such information in their research.

6.1.2 Education

As a part of the University of Minnesota, the ITS Laboratory in the Center for Transportation Studies has committed itself to promoting and advancing the education of transportation professionals. Since machine-vision technology is likely to be part of the future of traffic engineering, another MVL requirement is to provide an environment where machine-vision can be taught and learned interactively. This may include using the MVL as part of classroom exercises.

6.2 User Requirements

Different users will utilize the MVL in a number of ways. There are three types of uses that the MVL should support:

1. A single person using a single machine-vision device (with one or more camera feeds). A student or researcher examining a specific small number of camera views might use this configuration.

2. Multiple people, each using a single machine-vision device. This configuration would allow for multiple people to use machine-vision, as in a classroom setting.
3. A single person using multiple machine-vision devices. It is likely that a researcher will want as many camera feeds as possible, for wide-area detection and data collection. This configuration would give the user a maximum detection and data collection capability.

6.3 Available Resources

The MVL will utilize available resources where necessary or convenient. These include:

- The MVL will be integrated into the ITS Laboratory, and therefore should use the physical space offered by the ITS Lab (see Figure 2.2).
- The MVL will use the computing resources available in the ITS Lab.
- The MVL will utilize three machine-vision detection devices available in the ITS Lab.
- The MVL will use the video capabilities available in the ITS lab. This includes live feeds for up to thirteen I-394 cameras simultaneously, two University intersection camera feeds, and several VCRs for video taping and playback.

7 Machine-Vision Lab Design and Implementation

7.1 Facilities Layout

Figure 7.1 shows where the Machine-Vision Lab computers are located (indicated as LOC07, LOC09, and LOC10) in the ITS Lab. Notice the locations are ideally suited for an instructor to lead a classroom session from the front of the lab.

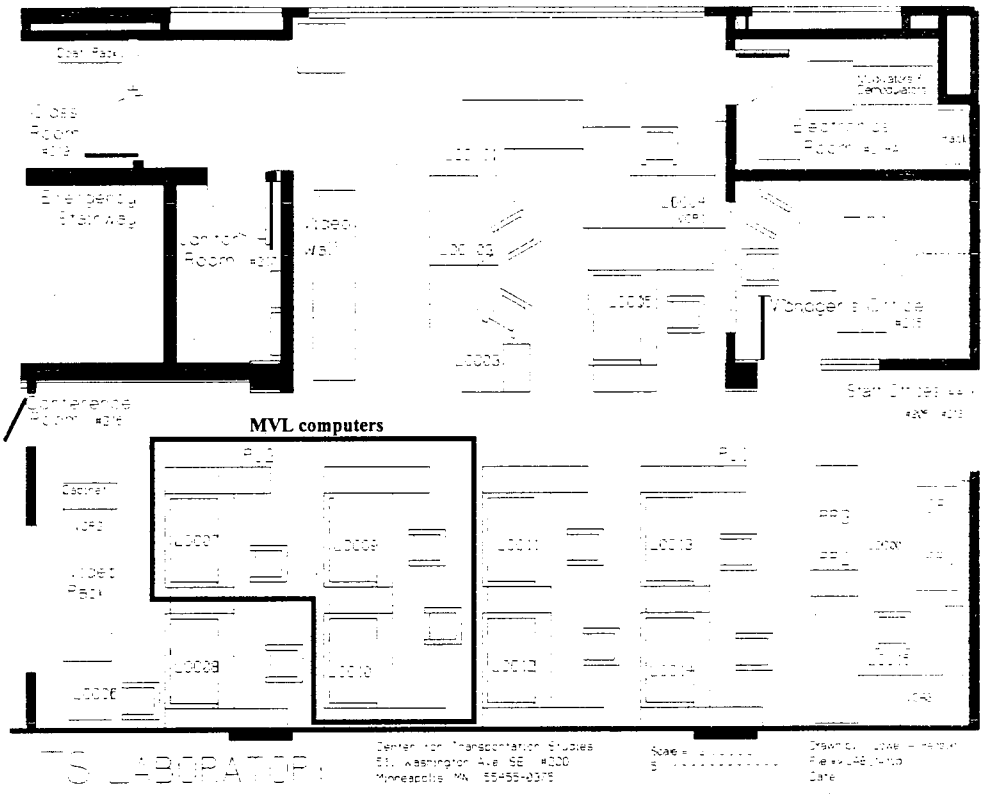


Figure 7.1 – MVL Workstation Locations

7.2 Workstation Configuration

Each workstation is equipped with the following hardware:

- Blue Star Pentium 90MHz PC with 17" monitor

- Hauppauge Motion Video Card
- TV monitor
- One or more VCRs
- Autoscope machine-vision device

Two of the workstations, Neptune (LOC07) and Orion (LOC09) have identical configurations, as is summarized in Figure 7.2.

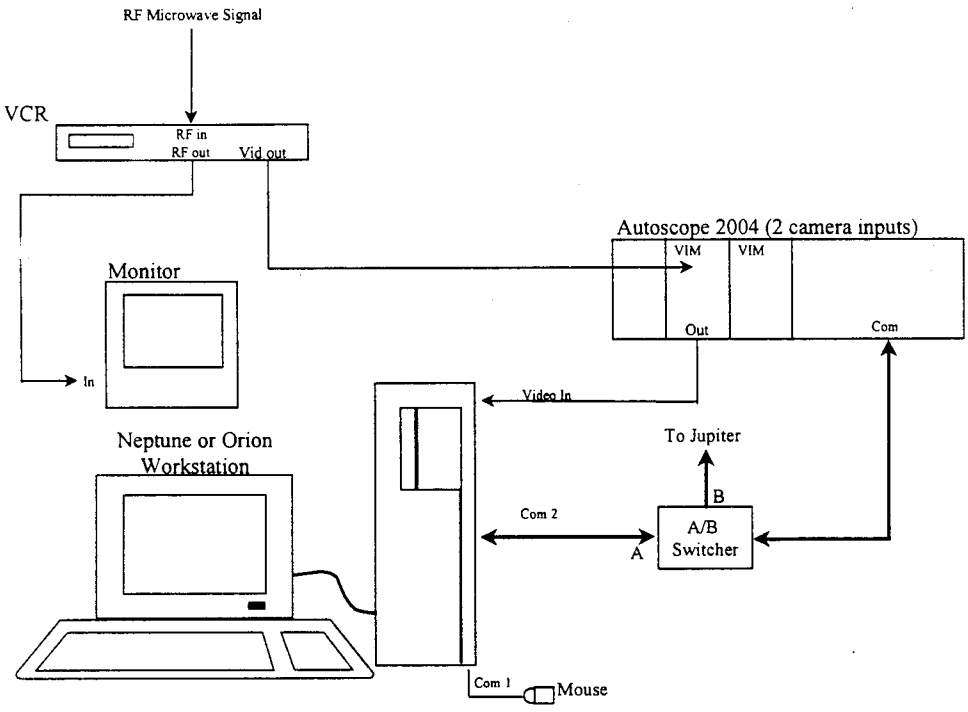


Figure 7.2 – Neptune or Orion Configuration

Each workstation has a live video input from the microwave video system in the ITS Lab. The VCR is used to tune into the desired signal, and can also be used for taping or playback. The TV monitor is useful for viewing the video signal being transmitted to the machine-vision device, but has no other purpose beyond that. The machine-vision device

(Autoscope), in conjunction with software on the PC, is used for set-up and execution of machine-vision detection on the video signal. The significance of the A/B switcher will be described momentarily.

The Jupiter (LOC10) workstation is configured slightly differently, as summarized in Figure 7.3.

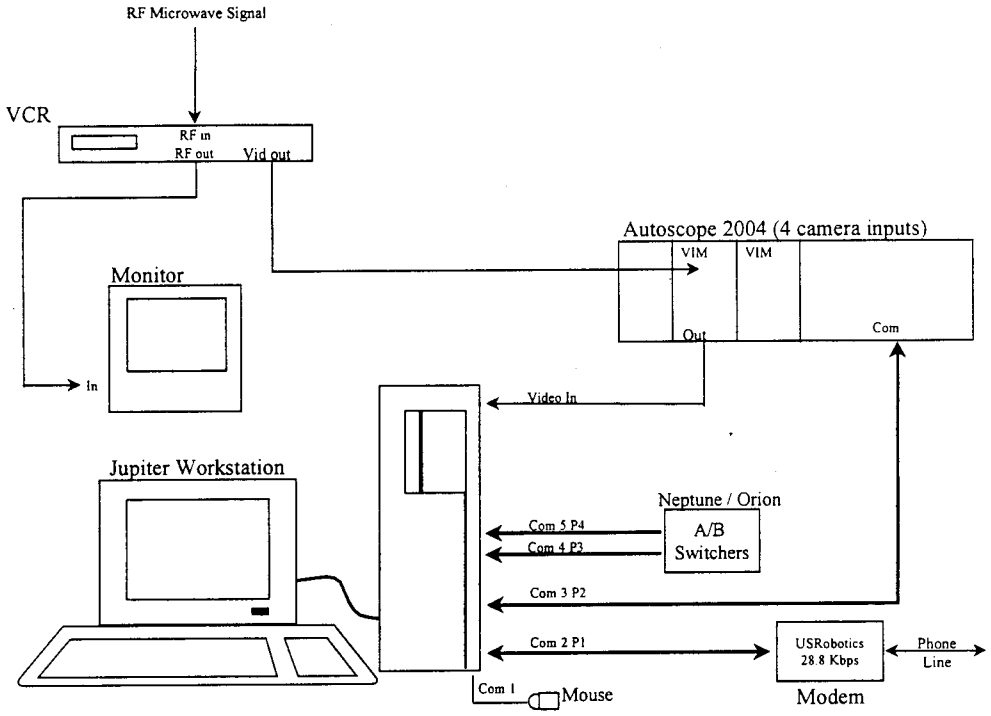


Figure 7.3 – Jupiter Configuration

The Jupiter workstation uses a serial port splitting device (DigiBoard) to allow for up to five simultaneous communication ports (one is always used by the mouse). One port is used for a modem, which has a connection to the TMC machine-vision computer using remote-control software (see the Phase 1 report for details), which is not part of the work in this project. Two A/B switchers are used for the option of connecting all three

machine-vision devices to the Jupiter workstation. Such a connection might be needed for a wide-area detection scheme where a researcher needed to utilize the capabilities of all the available MVL camera inputs. Figure 7.4 shows the complete workstation connection diagram.

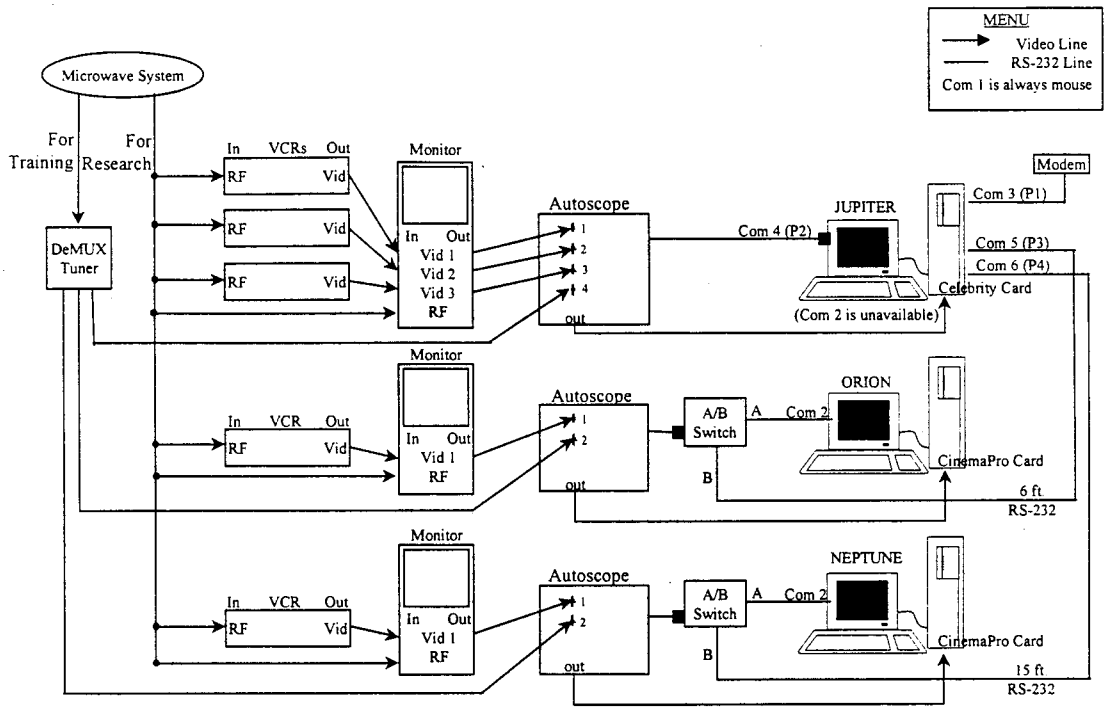


Figure 7.4 – MVL Design

Notice a few things about this design.

- When both A/B switchers are set on option A, the three workstations operate virtually identically. Each workstation has at least one camera signal which feeds through a VCR, allowing the user to tune into any desired camera signal (Jupiter has three such signals). Using one of these signals for machine-vision covers the first two user requirements (see Section 6.2).

- Each workstation has a camera input that comes from the same tuner device. This device would be controlled by an instructor/trainer, and would ensure that each workstation was seeing the same video in an education setting.
- By switching the A/B boxes to the B option, all three machine-vision devices are controllable from a single workstation (Jupiter). This is to accommodate the third user requirement (see Section 6.2), where a researcher would want to use as many camera feeds as possible.

7.3 Software Distribution

7.3.1 Video Drivers

- Jupiter workstation has a Hauppauge Celebrity video card and the video driver is dated on 6/11/96.
- Orion and Neptune workstations have the Hauppauge CinemaPro card and their video driver is dated on 2/26/97.

7.3.2 Autoscope Supervisor applications

- Supervisor application is used to create a detector file and its driver (asp.exe) is dated on 1/17/97.
- The Traffic Data Formatter is used for data collection and its driver (asfilter.exe) is dated on 1/15/97.

7.3.3 Autoscope ScopeServer applications

- ScopeServer for Windows32 is used to monitor the communication between the ScopeServer software and the Autoscope units. Its driver (ascserve.exe) is dated 12/1/97.
- Traffic Data Archiver TCP-IP is used for data collection from the Autoscope units and its driver (arch.exe) is dated 12/1/97.

7.4 Data Collection

Data can be collected from each of the workstations and corresponding machine-vision device by using the Autoscope Supervisor software located on each PC (this software is not describes in this report). To set-up a data collection session, the user has to set-up detectors and create a detector file as described on the Autoscope Supervisor User's Guide (see Figure 7.5).

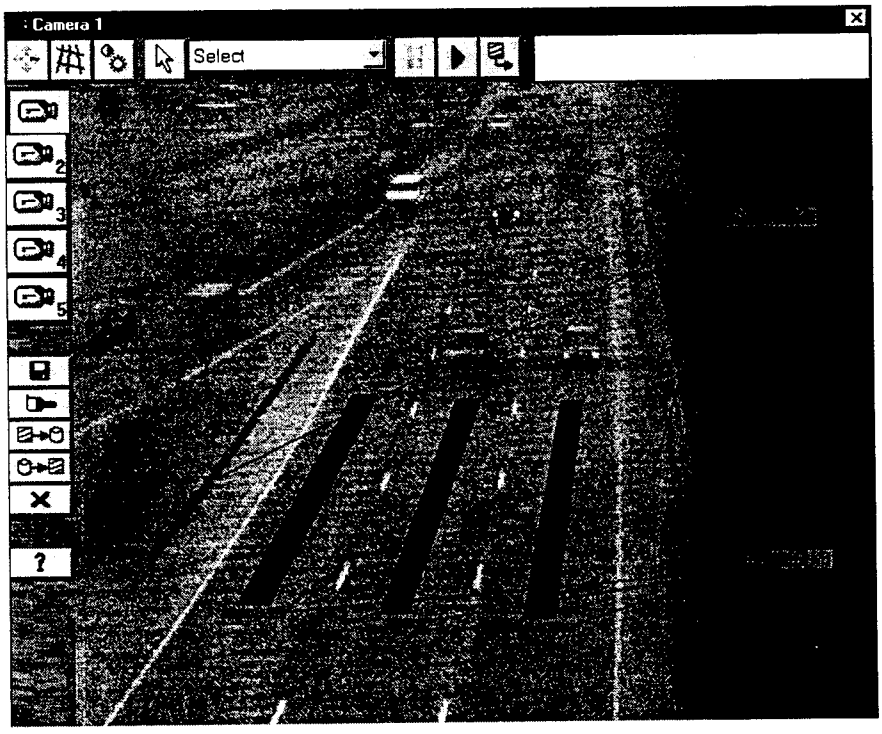


Figure 7.5 – Setting Detectors Using Autoscope Supervisor

When using the Jupiter workstation to collect data from multiple workstations simultaneously the Autoscope ScopeServer applications must be used. First, run ScopeServer for Windows 32. This opens a window to monitor all the activities between the ScopeServer computer (Jupiter) and the three Autoscope units. Second, establish communications between the ScopeServer computer and the three Autoscope units. Third, set the data collection time intervals and select detectors from the list of available detectors being used in each Autoscope (see Figure 7.6). These steps are discussed in more detail in Appendix F, and in the ScopeServer User's Guide.

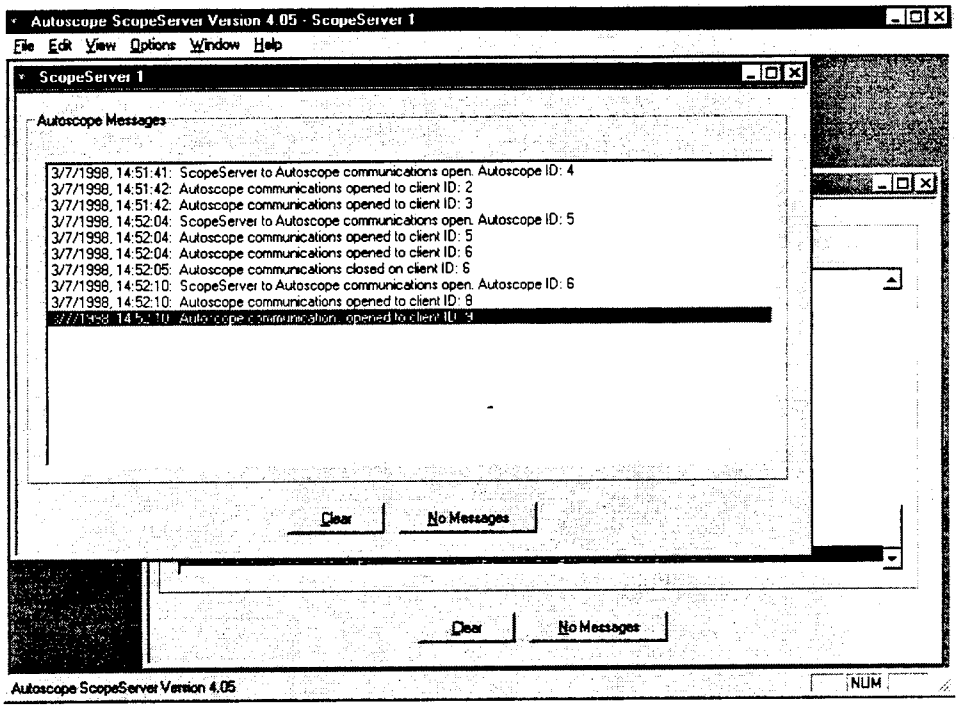


Figure 7.6 – Data Collection Using Autoscope ScopeServer

8 Conclusion

8.1 Results

The design of a Traffic Data Management System for the ITS Laboratory, combined with the integration of work from other projects and related work, should provide a useful backbone for traffic data management. The specific work of creating a Real-time Data Distributor further meets the objectives of this research by providing straightforward access to live loop sensor data across various platforms, operating systems, and types of software applications (like automated simulation, incident detection/management, and ramp metering systems). In addition, the use of a standard protocol and an open distributed system anticipates future growth, and provides a working a deployable information system for the next-generation traffic management center. Finally, the creation of a Machine-Vision Laboratory for education and research will enhance the use of new and advanced data sensing and collection techniques in ITS.

8.2 Recommendations

The Real-time Data Distributor could be expanded to provide other types of real-time data for ITS research and operations. For example, the TMC DDS is already capable of transmitting ramp-metering information. This data could easily be added to the RDD multicasting capabilities. Another type of live data that may be added in the future is incident data. At this time the ITS Laboratory does not have a mechanism for receiving this information. Indeed, any real-time data that can be packaged as an "object" could be handled by the RDD design offered in this research.

The Machine-Vision Lab will only be as good as the video signals it is getting. To date, there has not been a significant improvement in the quality of the video signals being sent to the ITS Lab from I-394. This needs to change.

8.3 Future Direction

The proposed upcoming phases of this research will concentrate on integrating work completed as part of programmatic funding efforts in the Center for Transportation Studies. In particular, research work in traffic data (as described in this report) will be combined with other work being completed in traffic simulation and traffic control. This mixture of work could result in the creation of a real-time automated simulation and control system. Such a system would provide a seamless integration of

- geometry data,
- real-time sensor data,
- historical sensor data,
- control scenarios,
- microscopic simulation, and
- macroscopic simulation

for use in real-time traffic management activities.

Appendix A - ITS Deployment Standards

Despite the many software and hardware tools available for traffic management, surveillance, and control, existing traffic management centers often do not capitalize on these tools. Such tools include, but are not limited to, ramp metering, simulation, incident detection and management, wide-area video detection (machine-vision), video surveillance, advanced data collection and management, variable message sign (VMS), geographic information (GIS) systems, and others. What is lacking is the integration of both software and hardware in a single workstation - for improved human factors design, efficiency, and effectiveness. Where in use, software applications most often run as stand-alone systems with minimal effectiveness. It is well recognized that in order to realize their full potential in traffic management, these tools must be integrated together to share both information and functionalities. In fact, the integration of software components is essential to the large-scale deployment of Advanced Traffic Management Systems (ATMS) in the Next-Generation Traffic Management Center (NGTMC). Until recently, there have been two significant obstacles preventing effective software integration in ATMS systems: 1) the working components of ITS had not been adequately defined; and 2) there were no standards by which these components could be properly integrated. However, two current initiatives are attempting to rectify this situation: the National ITS Architecture and the National Transportation Communications for ITS Protocol (NTCIP). Both entities have similar objectives: to maximize the potential for ITS system integration, interoperability, flexibility, and scaleability by developing standard definitions and protocols for design and deployment. A brief

examination of these two efforts will help illuminate the present state of software integration in ITS.

National ITS Architecture

A program to define and develop a National ITS Architecture was initiated in 1993 by the US Department of Transportation (USDOT). The objective in specifying such an architecture was to "facilitate the application of current and future technologies to improve the personal transportation experience and to improve the processes involved in moving people, goods, services, and information throughout the country." Additionally, ITS researchers, practitioners, and developers will benefit greatly from the enabling of system integration, common information structures, and open interface standards at many levels.

The National ITS Architecture provides a framework around which ITS systems may be defined, without actually specifying a system design or a design concept. The framework, as it stands today, can be subdivided into six related objectives:

1. Identification of the *technologies* (user services) to be facilitated by the architecture (e.g. traffic control or incident management),
2. Definition of the *operations* (functions) that must be performed for the implementation of each technology (e.g. gather traffic information or update a VMS),
3. Identification of the physical *subsystems* (entities or components) where the operations will be implemented (e.g. traffic management center or emergency provider),

Definition of the *interfaces* (information transfers) between the subsystems (e.g. traffic management center sends location/nature of an incident to emergency providers),

4. Specification of the *communication requirements* for the interfaces (e.g. internet or wireless), and
5. Identification and specification of *standards* necessary to promote national/regional interoperability and economy of scale considerations in product deployment.

The National ITS Architecture is an ongoing and evolving work, with the potential to "de-fragment" the current ITS program by getting everyone on the same page, so to speak. It also fosters a mindset that stresses integration and the potential for future expansion. Its greatest contribution to the specific problem of ATMS software integration is in providing a set of standard interface definitions, which at some level may be implemented as software applications sharing information. The importance of standard interfaces will become clearer later in this discussion. The National ITS Architecture, however, does not specify implementation standards, so the question of *how* to implement software integration remains.

National Transportation Communications for ITS Protocol (NTCIP)

Beginning in 1992, the National Electrical Manufacturers Association (NEMA), and later the Federal Highway Administration (FHWA), sponsored a series of discussions aimed at developing standard communication guidelines for traffic signal controllers. These discussions produced the first version of the National Transportation Communications for ITS Protocol (NTCIP) in December 1995, which defined a standard protocol for traffic

signal systems. Since that time, the NTCIP Joint Standards Committee has grown in both size and scope, and has extended its family of protocols to cover other electronic devices, such as computers, variable message signs, closed-circuit television cameras, freeway ramp meters, highway advisory radio, environmental sensor stations, and others. The aims of NTCIP are to 1) enable interoperability and interchangeability between disparate ITS devices and systems; and 2) expand user choice and flexibility in designing and deploying ITS systems. A communications protocol is a set of rules for message coding and transmission between devices. Unlike the national architecture, NTCIP is concerned with *how* information is transmitted, and therefore has more potential for directly aiding the goals of software integration in traffic management. However, the NTCIP committee has only recently begun identifying protocols for "center-to-center" (i.e. computer-to-computer) communications (referred to as "Class E" communications).

1 The center-to-center protocol being defined by the NTCIP committee is "intended to enable continuous and automated exchange of information between transportation management centers, or more specifically, between computer systems involved in real-time transportation management." The working group assigned the task of specifying the protocol were caught for some time between too rival protocol candidates: the CORBA approach, which uses the Common Object Request Broker Architecture (CORBA), and the Sockets approach, which uses TCP/IP sockets directly. Both approaches have advantages and disadvantages, and in the end it was decided that both approaches needed to be accommodated to cover the wide range of user needs present. Some sort of comprehensive integrated approach is now being worked on. This paper will not contribute to this discussion, but will instead

describe one of these approaches (CORBA) and show how it was used to solve a specific ITS software integration problem.

Appendix B - CORBA and Software Integration

The scenario we faced in meeting the requirements of a real-time data distributor (RDD) is a microcosm of a more general problem facing the software industry today. Due to advances in client-server computing and the internet, much of the software written today must conform to a broader set of requirements than it has in the past (#). Specifically:

1. It must run on a distributed network of machines, each individual component serving to meet some central goal of the software system,
2. Each machine may run a different operating system, such as UNIX, Windows, or Macintosh,
3. Different system components may be written in different programming languages, such as C, C++, Java, or Visual Basic,
4. The software components must be easy to integrate into future, perhaps unknown, systems, and
5. The components must be integrateable with existing *legacy* systems.

Software could perhaps be written to meet some of these requirements (using sockets and Remote Procedure Calls (RPCs), for example); however, the accomplishment would come at a great expense of time and effort, and would probably not be very flexible or scaleable. In particular, a solution is required which adheres to a set of standards that guarantee integration and future scalability.

The requirements are further complicated by the exponential growth of software interfaces. To illustrate this, consider a system with three software components (ramp metering, traffic simulation, and live sensor system, for instance) which need integrating

into a new traffic management tool while maintaining the uniqueness and integrity of each individual component. (Perhaps they were written in different locations, using different programming languages and running on different operating systems.) The programmer would have to write three dynamic software connections (interfaces) for complete interoperability, as shown in Figure A. 1. A fourth and fifth component raises the number of connections to six and ten, respectively.

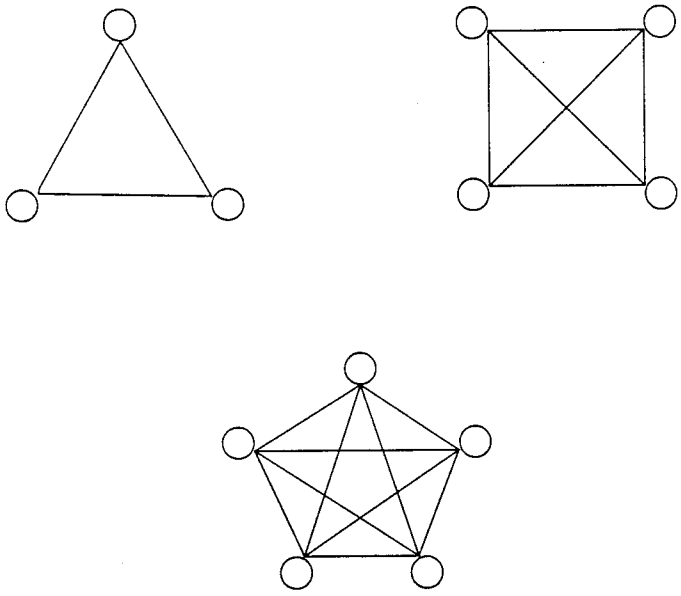


Figure A. 1 – Three, Four, and Five Components and Connections

Figure A. 2 illustrates a system of ten software components which requires 45 separate customized connections, each one possibly complex and fragile. The number of connections as a function of the number of components, n, is $(n*(n-1))/2$. This exponential increase in component interfaces is a serious drawback for most existing distributed programming and computing techniques.

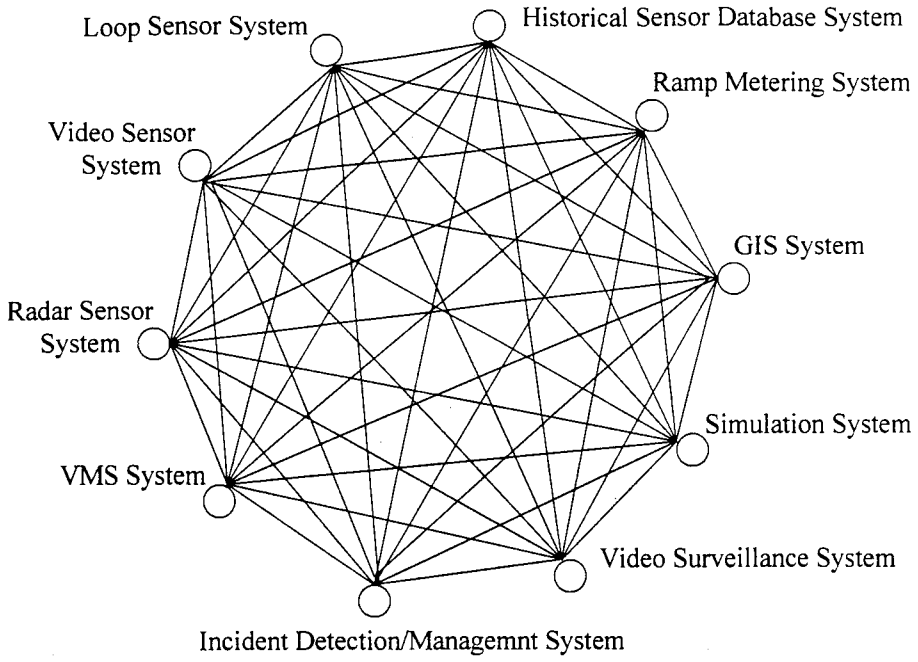


Figure A. 2 – Ten Components and Connections

The Common Object Request Broker Architecture (CORBA)

In the early 1990's, many of the world's major computer companies gathered together in a consortium, known as *the Object Management Group (OMG)*, to attempt to determine a set of standards which would meet the distributed integration requirements of the software industry. (Today, the OMG has over 700 company representatives.) The resulting, and constantly evolving, set of standards is called *the Common Object Request Broker Architecture (CORBA)*. CORBA has the potential to revolutionize the computing industry, and other industries along with it.

Software Objects

CORBA relies on object-oriented software methodologies, which will not be discussed in detail here. At the heart of object-orientation is, not surprisingly, the software *object*. An

object is a set of information that describes a logical entity as completely as possible. This description includes both the object's characteristics (*attributes*) and the functions (*methods*) which may be executed on it. As an example, a simplified *ramp_meter* object may have two attributes: an *indentification_number* and a *metering_rate*. In addition, the *ramp_meter* object may have two methods: *set_rate*, which sets the metering rate to a specified value, and *get_rate*, which returns the current metering rate. These four pieces of information describe the *ramp_meter* object's state and behavior.

There are a number of programming languages widely in use today which take advantage of object-oriented techniques to one degree or another, including C++, Smalltalk, Visual Basic, and Java. One of the attractive qualities of object-oriented programming is that objects seem to describe real-world entities much more closely. It has also proven to be a very useful and intuitive method of building and analyzing *interfaces* (software-to-user and software-to-software interfaces). These advantages, plus some advanced features of object-orientation like encapsulation, inheritance, polymorphism, and complexity hiding (#), make objects a powerful foundation for the CORBA protocol. With CORBA, software components dynamically inter-operate by sharing software objects, even across networks.

Object Request Brokers

At the heart of the CORBA standard is the *Object Request Broker*, or *ORB*. The term *broker* should be taken quite literally, for the ORB behaves like a broker of objects between different software components. In this way it manages the interactions of different software components in a centrally located place, much like a stockbroker would work for many investment clients. Thus, the exponential increase in software

interfaces experienced earlier can be avoided. Instead of 45 interfaces, ten components now have only *ten* interfaces (see Figure A. 3). Any software component physically connected (via a network) with another software component can interact with that component, provided that both components have knowledge of each other's interface specifications.

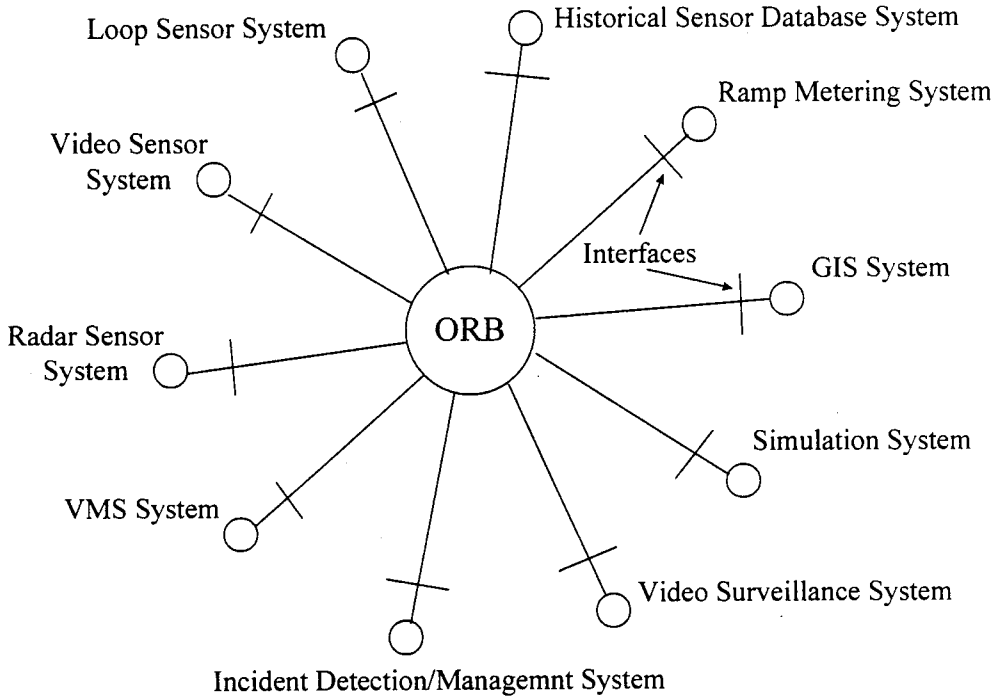


Figure A. 3 - Ten Components using an ORB

It is important to understand that CORBA is neither a programming language nor a software application. Rather, CORBA is a set of specifications, or standards, for how an ORB should work. Specific ORBs exist as proprietary software, and are produced by a number of companies (Orbix from Iona, NEO from Sun, ORB plus from HP, DSOM from IBM, Object Broker from DEC, Orbeline from PostModern, and PowerBroker from ExperSoft). It is the responsibility of these software vendors to ensure that their products

are "CORBA compliant." Because an ORB behaves as a "middleman" between software components, this technology is often referred to as *middleware*.(#)

Interface Definition Language

The design of interfaces represents the principal task for a programmer using the CORBA standard. Fortunately, the interface definition work of the National ITS Architecture and forthcoming work in NTCIP should help standardize these interface descriptions. Although CORBA is not a programming language, it does use a language known as the *Interface Definition Language (IDL)* to describe the interfaces that exist for different software components. An object's IDL describes how that software object can be viewed and/or used by any other software object. For example, our *ramp_meter* object described earlier could be defined in IDL as:

```
interface ramp_meter {
        readonly attribute long   identification_number;
        readonly attribute float  metering_rate;
        void                      set_rate (in float new_rate);
        float                     get_rate();
};
```

An experienced reader will notice that IDL looks and feels much like C++. Remember, however, that IDL is not a full programming language, in that it would not be used to actually execute any functions, but instead describes *how* the objects can be used by defining the object attributes and methods necessary for invocation. IDL is then used in

tandem with some other programming language (like C++) and an ORB to actually implement the functionality.

Because IDL has been created as a specific and compilable standard, the potential for smoothing ATMS software integration has been increased dramatically. In fact, if it hasn't done so already, the NTCIP Joint Standards Committee should plan to specify and publish a common library of standard ITS IDL interfaces for model development and deployment. In effect, by using a standard description for each software component's interface, and by making that interface transparent to other components in a network, the distributed system is opened up for interoperability.

Appendix C - Station-Detector Table

Loc. Type	Station	Right Lane Det	2 Det	3 Det	4 Det	5 Det	FROM	GIS Route ID	Dir	Milemark	Center Line Station
ML	1	176	177					MN65-D	SB	0.921	
EN	2	178	179	180				I35W-D	SB	16.252	
ML	2	178	179	180				MN65-D	SB	0.470	314+50
ML	3	181	182					I35W-D	SB	16.604	314+80
ML	4	183	184	185	186			I35W-D	SB	16.116	288+20
ML	5	187	188	189	190			I35W-D	SB	15.832	273+00
ML	6	191	192	193	194			I35W-D	SB	15.440	**
ML	7	195	196	197	198			I35W-D	SB	15.002	229+80
ML	8	199	200	201	202			I35W-D	SB	14.581	207+20
ML	9	203	204	205	206			I35W-D	SB	14.140	183+60
ML	10	207	208	209				I35W-D	SB	13.642	157+40
ML	11	210	211	212				I35W-D	SB	13.173	132+20
ML	12	213	214	215				I35W-D	SB	12.751	110+80
ML	13	571	572					I35W-D	SB	8.650	292+10
HL	13	573						I35W-D	SB	8.650	
ML	14	219	220	221				I35W-D	SB	12.340	90+70
ML	15	222	223	224				I35W-D	SB	11.879	487+90
ML	16	225	226					I35W-D	SB	11.675	**
ML	17	227	228	229				I35W-D	SB	11.129	444+70
ML	18	230	231					I35W-D	SB	10.720	420+40
ML	19	232	233					I35W-D	SB	10.326	402+70
ML	20	234	235					I35W-D	SB	9.763	374+20
ML	21	236	237					I35W-D	SB	9.364	354+70
ML	22	238	239					I35W-D	SB	9.100	
HL	22	351						I35W-D	SB	9.100	
ML	23	240	241					I35W-D	SB	8.800	313+60
HL	23	354						I35W-D	SB	8.800	
ML	24	242	243					I35W-D	SB	8.190	269+60
HL	24	355						I35W-D	SB	8.190	
ML	25	244	245					I35W-D	SB	7.760	241+20
HL	25	582						I35W-D	SB	7.760	
ML	26	246	247					I35W-D	SB	7.230	212+70
HL	26	583						I35W-D	SB	7.230	
ML	27	248	249					I35W-D	SB	6.670	184+20
HL	27	999						I35W-D	SB	6.670	
ML	28	250	251					I35W-D	SB	6.130	160+80
HL	28	1000						I35W-D	SB	6.130	
ML	29	252	253					I35W-D	SB	5.710	134+30
HL	29	1001						I35W-D	SB	5.710	
ML	30	254	255					I35W-D	SB	5.200	118+70
HL	30	1002						I35W-D	SB	5.200	
ML	31	256	257					I35W-D	SB	4.940	86+70
HL	31	1003						I35W-D	SB	4.940	
ML	32	258	259					I35W-I	NB	2.240	1837+20
HL	32	461						I35W-I	NB	2.240	
ML	33	260	261					I35W-I	NB	2.590	1856+70
HL	33	480						I35W-I	NB	2.590	
ML	35	264	265					I35W-I	NB	3.290	1893+40
HL	35	500						I35W-I	NB	3.290	
ML	36	266	267	525				I35W-I	NB	4.490	118+10
HL	36	1038						I35W-I	NB	4.490	
ML	37	268	269	270				I35W-I	NB	5.310	141+40

HL	37	533						I35W-I	NB	5.310	
ML	38	271	272					I35W-I	NB	5.720	161+30
HL	38	541						I35W-I	NB	5.720	
ML	39	273	274					I35W-I	NB	6.290	192+60
HL	39	545						I35W-I	NB	6.290	
ML	40	275	276					I35W-I	NB	6.680	219+40
HL	40	577						I35W-I	NB	6.680	
ML	41	277	278					I35W-I	NB	7.380	249+30
HL	41	579						I35W-I	NB	7.380	
ML	42	279	280					I35W-I	NB	7.760	269+60
HL	42	591						I35W-I	NB	7.760	
ML	43	281	282					I35W-I	NB	8.370	
HL	43	729						I35W-I	NB	8.370	
ML	44	283	284					I35W-I	NB	8.670	317+40
HL	44	350						I35W-I	NB	8.670	
ML	45	285	286					I35W-I	NB	9.013	335+50
ML	46	287	288					I35W-I	NB	9.364	354+70
ML	47	289	290					I35W-I	NB	9.763	374+20
ML	48	291	292					I35W-I	NB	10.326	402+70
ML	49	293	294					I35W-I	NB	10.720	420+40
ML	50	295	296	297				I35W-I	NB	11.125	444+90
ML	51	298	299					I35W-I	NB	11.537	468+90
ML	52	300	301	302				I35W-I	NB	11.879	487+90
ML	53	303	304	305				I35W-I	NB	12.386	91+70
ML	54	306	307	308				I35W-I	NB	12.696	109+70
ML	55	309	310	311				I35W-I	NB	13.173	132+20
ML	56	312	313	314				I35W-I	NB	13.630	156+80
ML	57	315	316	317	318			I35W-I	NB	14.121	183+20
ML	58	319	320	321	322			I35W-I	NB	14.581	207+20
ML	59	323	324	325	326			I35W-I	NB	15.005	229+80
ML	60	327	328	329	330			I35W-I	NB	15.440	
ML	61	331	332	333	334			I35W-I	NB	15.832	273+00
ML	62	335	336	337	338			I35W-I	NB	16.116	288+20
ML	63	339	340					I35W-I	NB	16.604	314+80
EX	64	341	342	343				I35W-I	NB	16.366	
ML	64	341	342	343				MN65-I	NB	0.470	314+50
ML	65	344	345					MN65-I	NB	0.921	
ML	66	1058	1059					MN62-I	EB	109.430	90+00
ML	67	1062	1063					MN62-I	EB	110.010	110+50
ML	68	1066	1067					MN62-I	EB	110.550	
ML	69	352	353					MN62-I	EB	111.100	
EN	70	124						I35W-I	NB	10.973	
ML	70	124						MN62-I	EB	111.365	
ML	71	359	360					I35W-I	NB	0.620	1751+30
ML	72	131	132					I35W-I	NB	1.660	1805+30
ML	73	346	347					TH13	EB		
ML	74	348	349					TH13	WB		
ML	75	141	142					MN62-D	WB	111.220	
ML	76	356	357	358				I94-D	WB	232.930	
ML	77	494	495					I35W-I	NB	4.270	86+70
HL	77	496						I35W-I	NB	4.270	
ML	78	1006	1007					I35W-D	SB	4.270	1893+40
HL	78	1008						I35W-D	SB	4.270	
ML	79	1011	1012					I35W-D	SB	3.290	1856+70
HL	79	1013						I35W-D	SB	3.290	
ML	80	8	9	10				I35W-D	SB	2.590	1837+20
HL	80	10						I35W-D	SB	2.590	
ML	81	1014	1015					I35W-D	SB	2.260	1805+30

HL	81	1016						I35W-D	SB	2.260	
ML	82	133	134					I35W-D	SB	1.660	1751+70
HL	82	161						I35W-D	SB	1.660	
ML	85	361	362	363	364			I94-D	WB	232.680	
ML	86	365	366	367	368			I94-D	WB	232.540	
ML	87	369	370	371	372			I94-D	WB	232.390	
ML	88	375	376	377				I94-D	WB	232.170	509+50
ML	89	378	379	380				I94-D	WB	232.140	504+00
ML	90	381	382	383				I94-D	WB	232.070	
ML	91	390	391	392				I94-D	WB	231.870	
ML	92	396	397	398				I94-D	WB	231.790	
ML	93	399	400	401				I94-D	WB	231.670	
EN	97	1897	1898	1899				I35E-D	SB	107.621	
ML	97	1897	1898	1899				I94-D	WB	242.290	
EX	101							MN62-I	EB	111.052	
ML	102	489	490	491				I94-D	WB	231.960	
ML	103	407	408	409				I94-I	EB	231.670	
ML	104	410	411	412				I94-I	EB	231.790	
ML	105	416	417	418				I94-I	EB	231.870	
ML	106	422	423	424				I94-I	EB	231.960	
ML	107	425	426	427				I94-I	EB	232.070	
ML	108	431	432	433				I94-I	EB	232.170	509+50
ML	109	413	414	415	419			I94-I	EB	232.540	
ML	110	497	498	499				I94-I	EB	232.930	
ML	119	465	466	467				I494-I	WB	5.440	**
ML	120	462	463	464				I494-D	EB	5.440	**
ML	122	617	618	619				I94-D	WB	231.270	
ML	123	621	622	623	624	625		I94-D	WB	230.762	224+40
ML	124	626	627	628	629	630		I94-D	WB	230.462	241+20
ML	125	631	632	633	634			I94-D	WB	229.600	282+60
ML	126	637	638	639	640			I94-D	WB	226.461	354+00
EX	127	1080	1081					I35W-D	SB	10.800	
ML	127	1080	1081					MN62-D	WB	111.100	
ML	128	649	650	651	652			I94-D	WB	223.937	233+20
ML	129	653	654	655				I94-D	WB	222.504	1158+20
ML	130	657	658	659				I94-I	EB	222.560	
ML	131	661	662					I694-I	EB	34.770	
EX	131	661	662					I94-I	EB	223.197	
ML	132	1083	1084					MN62-D	WB	110.550	
ML	134	681	682					I694-I	EB	35.260	1256+60
ML	135	687	688	689	690			I94-I	EB	226.461	354+00
ML	136	696	697	698	699			I94-I	EB	227.887	278+70
ML	137	703	704	705	706			I94-I	EB	229.657	282+60
ML	138	711	712	713	714	715		I94-I	EB	230.462	241+20
ML	139	719	720	721				I94-I	EB	230.762	224+40
ML	140	726	727	728				I94-I	EB	231.290	
ML	141	730	731					I94-I	EB	231.400	
ML	142	521	522	523	524			I694-I	EB	35.760	1277+90
ML	143	436	437	438				I694-D	WB	35.760	1278+00
ML	144	443	444	445	446			I694-D	WB	35.910	1286+00
ML	145	439	440	441	442			I694-I	EB	35.910	1286+00
ML	146	665	666	667	668			I94-I	EB	223.937	233+20
ML	147	393	394	395				I694-I	EB	36.600	
ML	148	515	516	517				I694-D	WB	36.600	
ML	149	526	527	528	529			I694-I	EB	36.310	
ML	150	402	403	404	405			I694-D	WB	37.300	
ML	151	501	502	503				I694-D	WB	38.690	432+20
ML	152	504	505	506				I694-D	WB	38.150	403+60

ML	153	734	735	736				I694-D	WB	37.880	389+80
ML	154	587	588	589				I694-D	WB	37.530	369+60
ML	156	512	513	514				I694-D	WB	36.970	340+90
ML	158	518	519	520				I694-D	WB	36.310	
ML	159	387	388					I94-D	WB	224.912	279+40
ML	160	534	535	536	537			I694-I	EB	37.300	
ML	163	530	531	532				I694-I	EB	36.960	340+60
ML	165	538	539	540				I694-I	EB	37.520	369+30
ML	166	542	543	544				I694-I	EB	37.860	389+60
EX	166							MN62-I	EB	111.362	
ML	167	546	547	548				I694-I	EB	38.140	403+30
ML	168	549	550	551				I694-I	EB	38.690	431+90
ML	170	599	600					I94-I	EB	224.912	279+40
ML	171	741	742	743				I694-I	EB	38.900	
ML	172	744	745	746				I694-D	WB	38.900	
ML	173	747	748	749				I694-I	EB	39.200	457+10
ML	174	750	751	752				I694-D	WB	39.200	457+40
ML	175	757	758	759				I694-I	EB	39.620	478+10
ML	176	760	761	762				I694-D	WB	39.610	477+80
ML	177	17	18	19				I694-I	EB	39.980	
ML	178	14	15	16				I694-D	WB	39.980	
ML	179	428	429	430				I694-I	EB	40.330	517+30
ML	180	216	217	218				I694-D	WB	40.340	517+80
ML	181	888	889	890				I494-I	WB	3.748	814+80
ML	182	878	879	880				I494-I	WB	4.345	184+10
ML	183	868	869	870				I494-I	WB	4.836	758+10
ML	184	456	457					I694-I	EB	40.700	536+70
ML	185	854	855	856				I494-I	WB	5.822	705+40
ML	186	843	844	845				I494-I	WB	6.372	698+80
ML	187	826	827	828				I494-I	WB	6.665	653+50
ML	188	831	832	833				I494-I	WB	7.542	614+20
ML	189	815	816					I494-I	WB	7.859	598+20
ML	190	808	809	810				I494-I	WB	8.154	581+60
ML	191	801	802					I494-I	WB	8.587	559+30
ML	192	803	804					I494-D	EB	8.587	559+30
ML	193	811	812	813				I494-D	EB	8.154	581+60
ML	194	819	820	821				I494-D	EB	7.859	598+20
ML	195	834	835	836				I494-D	EB	7.542	614+20
ML	196	837	838	839				I494-D	EB	6.665	653+50
ML	197	846	847	848				I494-D	EB	6.372	698+80
ML	198	849	850	851				I494-D	EB	5.822	705+40
ML	199	614	615	616				I694-D	WB	40.690	536+80
ML	200	873	874	875				I494-D	EB	4.836	758+10
ML	201	883	884	885				I494-D	EB	4.345	784+10
ML	202	891	892	893				I494-D	EB	3.753	814+40
ML	203	864	865					I694-I	EB	40.900	547+70
ML	204	859	860					I694-D	WB	40.910	547+80
ML	205	900	901	902				I94-I	EB	215.614	2083+40
ML	206	903	904	905				I94-D	WB	215.614	2083+40
ML	207	907	908	909				I94-I	EB	216.131	2111+00
ML	208	910	911	912				I94-D	WB	216.131	2111+00
ML	209	913	914					I494-D	EB	27.776	
EX	209	913	914					I94-I	EB	216.500	
ML	211	917	918					I94-I	EB	216.720	2144+00
ML	212	919	920					I494-D	EB	27.663	
EX	212	919	920					I94-D	WB	216.700	
ML	213	921	922					I94-D	WB	216.694	2136+60
ML	215	925	926					I94-I	EB	217.570	900+00

ML	216	927	928	929				I94-D	WB	217.488	896+60
ML	217	934	935					I94-I	EB	218.306	937+60
ML	218	936	937					I94-D	WB	218.306	937+60
ML	219	938	939					I94-I	EB	218.890	968+30
ML	220	940	941					I94-D	WB	218.900	968+50
ML	221	991	992					US169-I	NB	137.660	776+80
ML	222	993	994					US169-D	SB	137.660	776+80
ML	223	950	951	952				I94-I	EB	219.650	
ML	224	953	954					I94-D	WB	219.650	
ML	225	959	960					I94-I	EB	220.000	
ML	226	961	962					I94-D	WB	220.030	
ML	227	963	964					I94-I	EB	220.559	1055+80
ML	228	965	966					I94-D	WB	220.559	1055+80
ML	229	971	972					I94-I	EB	221.359	1097+60
ML	230	973	974					I94-D	WB	221.359	1097+60
ML	231	976	977					I94-I	EB	222.130	
ML	232	978	979	980				I94-D	WB	222.130	
ML	233	983	984	985	986			I94-I	EB	223.132	1191+30
ML	234	987	988	989	990			I94-D	WB	223.132	1191+30
ML	235	685	686	691				I94-I	EB	224.440	260+60
ML	236	641	642					I94-D	WB	224.440	1259+80
ML	237	643	644					I694-D	WB	35.420	1259+80
EN	237	643	644					I94-D	WB	224.277	
EX	238	593	594					I94-D	WB	224.977	
EN	239	595	596					I94-I	EB	224.935	
ML	240	476	477	478	479			I94-I	EB	225.030	
ML	241	481	482	483	484			I94-D	WB	225.050	428+40
ML	242	468	469	470	471			I94-I	EB	225.470	405+70
ML	243	472	473	474	475			I94-D	WB	225.470	405+70
ML	244	507	508	509	510			I94-I	EB	225.964	380+30
ML	245	451	452	453	454			I94-D	WB	225.964	380+30
ML	246	601	602	603	604			I94-I	EB	226.780	336+20
ML	247	605	606	607	608			I94-D	WB	226.790	335+60
ML	248	86	87	88	89			I94-I	EB	227.150	317+60
ML	249	90	91	92	93			I94-D	WB	227.150	317+00
ML	250	82	83	84	85			I94-I	EB	227.490	229+50
ML	251	78	79	80	81			I94-D	WB	227.500	229+00
ML	252	73	74	75	76			I94-D	WB	227.887	278+70
ML	253	59	60	61	62	63		I94-I	EB	228.272	258+00
ML	254	65	66	67	68	69		I94-D	WB	228.272	258+00
ML	255	49	50	51	52	53		I94-I	EB	228.830	
ML	256	54	55	56	57	58		I94-D	WB	228.830	
ML	257	38	39	40	41	42		I94-I	EB	229.285	204+70
ML	258	43	44	45	46	47		I94-D	WB	229.285	204+70
ML	259	26	27	28	29	30		I94-D	WB	230.095	259+60
ML	260	21	22	23	24	25		I94-I	EB	230.095	259+60
ML	261	552	553					I394-D	WB	9.380	
ML	262	564	567					I394-D	WB	8.850	
ML	264	1631	1632	1633				US12-I	EB	155.020	
ML	265	384	385					I35W-D	SB	0.620	0.166
ML	266	1635	1636	1637				US12-I	EB	155.490	
ML	267	1641	1642	1643				US12-I	EB	156.290	
ML	268	1645	1646	1647				US12-I	EB	156.880	
ML	269	1651	1652	1653				I394-I	EB	0.290	980+10
ML	270	1656	1657	1658				I394-I	EB	0.850	1009+90
ML	271	1661	1662	1663				I394-I	EB	1.414	1040+00
ML	272	1669	1670	1671				I394-I	EB	2.020	1071+80
ML	273	1676	1677	1678	1679			I394-I	EB	2.510	1097+90

ML	274	1682	1683	1684				I394-I	EB	3.010	1124+30
ML	275	1686	1687	1688				I394-I	EB	3.520	1150+90
ML	276	1693	1694	1695	1696			I394-I	EB	3.879	1169+90
ML	276	1320	1321					MN5-I	EB	50.950	
ML	277	1698	1699	1700				I394-I	EB	4.368	1195+50
ML	278	1702	1703	1704	1705			I394-I	EB	4.884	1223+20
ML	279	1708	1709	1710				I394-I	EB	5.399	1250+20
ML	280	1712	1713					I394-I	EB	5.589	1271+20
ML	281	1719	1720	1721				I394-I	EB	6.149	1290+00
ML	282	1730	1731	1732				I394-I	EB	6.557	1322+80
ML	283	788	789	790				I394-D	WB	6.990	
ML	284	793	794	795				I394-I	EB	6.990	
ML	285	777	778	779				I394-D	WB	7.560	
ML	286	783	784	785				I394-I	EB	7.470	
ML	287	767	768	769				I394-D	WB	7.850	
ML	288	770	771	772				I394-I	EB	7.850	
ML	289	597	598					I394-D	WB	8.540	
ML	290	673	674					I394-I	EB	8.540	
ML	291	611	701	702				I394-I	EB	8.850	
ML	292	1866	1867					I494-D	EB	19.560	
ML	293	1869	1870					I494-D	EB	19.300	467+40
ML	294	1872	1873					I494-D	EB	18.860	436+70
ML	295	1874	1875					I494-D	EB	18.315	409+90
ML	296	1877	1878					I494-D	EB	17.513	396+50
ML	297	1880	1881					I494-D	EB	16.984	368+20
ML	298	1883	1884					I494-D	EB	16.218	327+20
ML	299	1888	1889					I494-D	EB	15.580	291+90
ML	301	1018	1019					MN62-I	EB	104.070	334+90
ML	302	1020	1021					MN62-I	EB	104.520	359+40
ML	303	1025	1026					MN62-I	EB	105.080	387+00
ML	304	1027	1028					MN62-I	EB	105.400	3975+30
ML	306	1322	1323					US212-I	NB	159.750	434+80
ML	307	1326	1327					US212-I	NB	160.340	467+20
ML	308	1329	1330					US212-I	NB	160.870	495+00
ML	309	1332	1333					US212-I	NB	161.290	516+90
ML	310	1029	1030					US212-I	NB	161.540	529+90
ML	311	1036	1037					MN62-I	EB	105.930	4003+60
ML	312	1034	1035					US212-I	NB	162.060	557+30
ML	313	1041	1042					MN62-I	EB	106.650	4041+80
ML	314	1044	1045					MN62-I	EB	107.300	4076+70
ML	315	1048	1049					MN62-I	EB	107.740	99+80
ML	316	1052	1053					MN62-I	EB	108.880	53+11
ML	317	1054	1055					MN62-I	EB	109.130	65+00
ML	318	1740	1741	1742				I394-D	WB	6.557	1322+80
ML	319	1722	1723					I394-D	WB	6.149	1290+00
ML	320	1750	1751					I394-D	WB	5.899	1271+20
ML	321	1754	1755	1756				I394-D	WB	5.399	1250+20
ML	322	1134	1135					MN62-I	EB	112.540	58+60
ML	323	1137	1138					MN62-I	EB	113.100	87+80
ML	324	1142	1143					MN62-I	EB	113.740	121+40
ML	325	1146	1147					MN62-I	EB	114.510	162+60
ML	326	1150	1151					MN62-I	EB	114.960	136+60
ML	327	1153	1154					MN62-I	EB	115.390	209+60
ML	330	1156	1157					MN62-D	WB	115.380	209+30
ML	331	1159	1160					MN62-D	WB	114.730	174+70
ML	332	1163	1164					MN62-D	WB	114.220	148+30
ML	333	1167	1168					MN62-D	WB	113.520	111+20
ML	334	1170	1171					MN62-D	WB	113.090	87+40

ML	335	1173	1174					MN62-D	WB	112.450	53+80
ML	336	1760	1761	1762	1763			I394-D	WB	4.884	1223+20
ML	337	1765	1766	1767				I394-D	WB	4.368	1195+50
ML	338	1771	1772	1773				I394-D	WB	3.879	1169+90
ML	339	1776	1777	1778				I394-D	WB	3.520	1150+90
ML	340	1780	1781	1782				I394-D	WB	3.010	1124+60
ML	341	1787	1788	1789	1790			I394-D	WB	2.510	1098+10
ML	342	1792	1793	1794				I394-D	WB	2.020	1071+80
ML	343	1797	1798	1799				I394-D	WB	1.414	1040+00
ML	344	1801	1802	1803				I394-D	WB	0.850	109+90
ML	345	1806	1807	1808				I394-D	WB	0.290	980+10
ML	346	1811	1812	1813				US12-D	WB	156.880	
ML	347	1820	1821	1822				US12-D	WB	156.290	
ML	348	1825	1826	1827				US12-D	WB	155.500	
ML	349	1830	1831	1832				US12-D	WB	155.020	
ML	350	1087	1088					MN62-D	WB	110.010	110+50
ML	351	1091	1092					MN62-D	WB	109.430	90+00
ML	352	1094	1095					MN62-D	WB	109.130	65+00
ML	353	1097	1098					MN62-D	WB	108.880	53+00
ML	354	1101	1102					MN62-D	WB	107.740	100+20
ML	355	1103	1104					MN62-D	WB	107.290	4076+20
ML	356	1107	1108	1109				MN62-D	WB	106.670	4042+50
ML	357	1112	1113					MN62-D	WB	105.930	4003+60
ML	358	1115	1116					US212-D	SB	162.060	557+30
ML	359	1122	1123					US212-D	SB	161.570	530+30
ML	360	1334	1335					US212-D	SB	161.290	516+90
ML	361	1337	1338					US212-D	SB	160.870	494+80
ML	362	1340	1341					US212-D	SB	160.340	467+20
ML	363	1344	1345					US212-D	SB	159.750	434+80
ML	364	1346	1347					US212-D	SB	159.560	
ML	366	1119	1120					MN62-D	WB	105.450	3979+80
ML	367	1124	1125					MN62-D	WB	105.090	387+20
ML	368	1127	1128					MN62-D	WB	104.530	359+90
ML	369	1130	1131					MN62-D	WB	104.080	335+20
ML	375	458	459	460				US100-I	NB	0.420	30+40
ML	376	1256	1257	1258				US100-I	NB	1.260	72+90
ML	377	1260	1261	1262				US100-I	NB	1.740	78+90
ML	378	1264	1265	1266				US100-I	NB	2.030	113+60
ML	379	1268	1269	1270				US100-I	NB	2.520	139+50
ML	380	1272	1273	1274				US100-I	NB	3.230	173+70
ML	381	1278	1279	1280				US100-I	NB	3.692	198+50
ML	382	2001	2002	2003				US100-I	NB	4.411	238+50
ML	383	2005	2006	2007				US100-I	NB	4.930	760+60
ML	384	2009	2010					US100-I	NB	5.317	784+00
ML	386	2020	2021					US100-I	NB	5.993	822+10
ML	387	2025	2026					US100-I	NB	6.670	855+80
ML	388	2027	2028					US100-I	NB	6.960	878+90
ML	389	2030	2031	2032				US100-I	NB	7.502	903+90
ML	391	2037	2038	2039				US100-I	NB	7.902	928+10
ML	392	2042	2043	2044				US100-I	NB	8.365	947+40
ML	393	2050	2051					US100-I	NB	9.001	128+20
ML	394	2053	2054					US100-I	NB	9.344	146+80
ML	395	2055	2056					US100-I	NB	10.017	181+40
ML	397	2061	2062					US100-D	SB	9.830	171+60
ML	398	2064	2065					US100-D	SB	9.336	146+40
ML	400	2068	2069					MN55-D	SB	186.500	
ML	401	2070	2071					MN55-I	NB	186.500	
ML	403	2073	2074					US100-D	SB	8.824	118+80

ML	404	2077	2078	2079				US100-D	SB	8.365	947+40
ML	405	2085	2086	2087				US100-D	SB	7.902	928+10
ML	407	2092	2093	2094				US100-D	SB	7.502	903+20
ML	408	2095	2096	2097				US100-D	SB	6.960	878+90
ML	409	2099	2100	2101				US100-D	SB	6.670	855+80
ML	410	2103	2104					US100-D	SB	6.099	828+50
ML	412	2114	2115	2116				US100-D	SB	5.308	783+50
ML	413	2119	2120	2121				US100-D	SB	4.886	760+20
ML	414	2123	2124	2125				US100-D	SB	4.411	238+50
ML	415	1282	1283	1284				US100-D	SB	3.681	197+90
ML	416	1286	1287	1288				US100-D	SB	3.220	173+70
ML	417	1290	1291	1292				US100-D	SB	2.520	139+50
ML	418	1294	1295	1296				US100-D	SB	2.020	114
ML	419	1298	1299	1300				US100-D	SB	1.740	78+90
ML	420	1302	1303	1304				US100-D	SB	1.250	72+40
ML	421	995	996					US100-D	SB	0.420	30+00
ML	426	1350	1351					US169-I	NB	122.580	304+50
ML	427	1353	1354					US169-I	NB	123.200	343+20
ML	428	1356	1357					US169-I	NB	123.520	360+40
ML	429	1359	1360					US169-I	NB	124.330	44+30
ML	430	1362	1363					US169-I	NB	125.090	79+40
ML	431	1902	1903					US169-I	NB	126.086	130+90
ML	432	1906	1907					US169-I	NB	126.920	176+20
ML	433	1910	1911					US169-I	NB	127.352	195+80
ML	434	1912	1913					US169-I	NB	127.660	214+10
ML	435	1915	1916					US169-I	NB	127.970	230+90
ML	437	1922	1923					US169-I	NB	128.371	163+70
ML	438	1926	1927					US169-I	NB	128.779	183+20
ML	439	1930	1931					US169-I	NB	129.391	214+70
ML	440	1933	1934					US169-I	NB	130.140	
ML	441	1936	1937					US169-I	NB	130.390	
ML	442	1942	1943					US169-I	NB	130.830	156+40
ML	443	1946	1947					US169-I	NB	131.050	428+40
ML	446	1955	1956					US169-D	SB	131.040	428+20
ML	447	1959	1960					US169-D	SB	130.820	156+20
ML	448	1964	1965					US169-D	SB	130.390	
ML	450	1967	1968					US169-D	SB	130.140	130.08
ML	451	1970	1971					US169-D	SB	129.383	214+30
ML	452	1973	1974					US169-D	SB	128.778	183+60
ML	453	1977	1978					US169-D	SB	128.240	152+50
ML	455	1985	1986					US169-D	SB	127.516	205+80
ML	456	1988	1989					US169-D	SB	127.343	195+80
ML	457	1992	1993					US169-D	SB	126.674	163+10
ML	458	1996	1997					US169-D	SB	125.838	118+90
ML	459	1366	1367					US169-D	SB	125.090	79+40
ML	460	1370	1371					US169-D	SB	124.300	44+30
ML	461	1373	1374					US169-D	SB	123.530	360+40
ML	462	1376	1377					US169-D	SB	123.210	343+50
ML	463	1379	1380					US169-D	SB	122.580	304+50
ML	464	1381	1382					US169-D	SB	122.300	296+10
ML	465	2611	2612	2613	2614			I94-I	EB	234.190	101+90
ML	466	2617	2618	2619				I94-I	EB	235.060	135+90
ML	467	2620	2621	2622				I94-I	EB	235.390	
ML	468	2625	2626	2627				I94-I	EB	236.490	
ML	469	2630	2631	2632	2633			I94-I	EB	237.260	
ML	470	1180	1181					I494-D	EB	14.893	257+40
ML	471	1183	1184					I494-D	EB	13.810	126+50
ML	472	1186	1187					I494-D	EB	12.890	151+50

ML	473	1188	1189	1190				I494-D	EB	12.420	201+10
ML	474	1195	1196					I494-D	EB	11.790	87+90
ML	475	1198	1199					I494-D	EB	11.200	57+40
ML	476	1202	1203					I494-D	EB	10.240	472+40
ML	477	1205	1206					I494-D	EB	9.570	508+10
ML	478	1207	1208					I494-D	EB	9.070	534+90
ML	479	2634	2635	2636				I94-I	EB	238.660	
ML	480	1209	1210					I494-I	WB	9.060	535+20
ML	481	1211	1212					I494-I	WB	9.570	508+10
ML	482	1214	1215					I494-I	WB	10.240	472+40
ML	483	1217	1218					I494-I	WB	11.200	57+40
ML	484	1224	1225					I494-I	WB	11.990	100+10
ML	485	1228	1229	1230				I494-I	WB	12.400	126+30
ML	486	1231	1232					I494-I	WB	12.890	151+20
ML	487	1234	1235					I494-I	WB	13.810	201+10
ML	488	1237	1238					I494-I	WB	14.893	257+40
ML	489	2638	2639	2640	2641			I94-I	EB	239.580	
ML	490	2643	2644	2645	2646			I94-I	EB	240.560	
ML	491	2648	2649	2650				I94-I	EB	241.230	
ML	492	1532	1533	1534				I494-D	EB	2.490	880+60
ML	493	1537	1538	1539	1540	1541		I494-D	EB	1.960	910+80
ML	494	1545	1546	1547				I494-D	EB	1.090	954+70
ML	495	1550	1551	1552	1553			MN5-I	EB	62.120	992+50
ML	496	1555	1556	1557				MN5-I	EB	62.580	1005+60
ML	497	1561	1562					MN5-I	EB	63.090	1033+50
ML	498	1565	1566					MN5-I	EB	63.690	1065+30
EN	499	2485	2486	2487				I35E-I	NB	107.360	
ML	499	2485	2486	2487				I94-I	EB	241.550	
EX	500	2490	2491	2492				I35E-D	SB	107.300	
ML	500	2490	2491	2492				I94-D	WB	241.560	
ML	501	1575	1576					MN5-D	WB	63.690	1065+80
ML	502	1581	1582					MN5-D	WB	63.140	1035+80
ML	503	1586	1587	1588				MN5-D	WB	62.400	996+90
ML	504	1590	1591	1592				MN5-D	WB	62.120	992+50
ML	505	1597	1598	1599				I494-I	WB	1.110	954+00
ML	506	1603	1604	1605	1606	1607		I494-I	WB	1.920	914+10
ML	507	1609	1610	1611				I494-I	WB	2.480	866+30
ML	508	1615	1616	1617				I494-I	WB	2.760	880+90
EX	509	1891	1892	1893				I35E-I	NB	107.630	
ML	509	1891	1892	1893				I94-I	EB	242.180	
ML	510	1517	1518					MN55-I	NB	198.040	
ML	511	1840	1841					I494-I	WB	15.580	291+90
ML	512	1843	1844					I494-I	WB	16.327	332+30
ML	513	1848	1849					I494-I	WB	16.994	368+70
ML	514	1851	1852					I494-I	WB	17.520	396+20
EN	515	1854	1855					I494-I	WB	18.305	409+60
ML	516	1856	1857					I494-I	WB	18.866	437+00
ML	517	1858	1859					I494-I	WB	19.310	467.4
ML	518	1861	1862					I494-I	WB	19.556	
ML	519	1527	1528					MN55-D	SB	198.040	
ML	520	1405	1406	1407				MN77-I	NB	6.820	1165+20
ML	521	1408	1409	1410				MN77-I	NB	7.180	1183+80
ML	522	1412	1413	1414				MN77-I	NB	7.690	1210+20
ML	523	1417	1418	1419	1420			MN77-I	NB	8.190	1237+90
ML	524	1423	1424	1425	1426			MN77-I	NB	8.500	1252+80
ML	525	1427	1428					MN77-I	NB	8.830	1270+30
ML	526	1436	1437					MN77-I	NB	9.330	1297+20
ML	527	1439	1440					MN77-I	NB	9.540	1308+30

ML	528	1442	1443					MN77-I	NB	10.010	1332+90
ML	529	1445	1446					MN77-I	NB	10.460	173+30
ML	530	1449	1450					MN77-I	NB	11.000	200+50
ML	531	1451	1452					MN77-I	NB	11.320	215+60
ML	532	1520	1521	1522				MN55-I	NB	197.710	
ML	533	1524	1525					MN55-D	SB	197.710	
ML	534	1467	1468					MN77-D	SB	11.320	215+60
ML	535	1470	1471					MN77-D	SB	11.000	200+50
ML	536	1472	1473					MN77-D	SB	10.450	173+00
ML	537	1476	1477					MN77-D	SB	10.000	1332+50
ML	538	1480	1481					MN77-D	SB	9.550	1308+60
ML	539	1485	1486					MN77-D	SB	9.340	1297+50
ML	540	1493	1494					MN77-D	SB	8.840	1270+60
ML	541	1496	1497					MN77-D	SB	8.500	1253+20
ML	542	1500	1501	1502				MN77-D	SB	8.190	1238+30
ML	543	1504	1505	1506				MN77-D	SB	7.680	1210+50
ML	544	1509	1510	1511				MN77-D	SB	7.170	1183+40
ML	545	1512	1513	1514				MN77-D	SB	6.820	1165+20
ML	546	2653	2654	2655				I94-D	WB	241.230	
ML	547	2659	2660	2661	2662			I94-D	WB	240.230	
ML	548	2663	2664	2665	2666			I94-D	WB	239.340	
ML	549	2668	2669	2670				I94-D	WB	238.300	
ML	550	2672	2673	2674	2675			I94-D	WB	236.930	
ML	551	2676	2677	2678				I94-D	WB	236.480	
ML	552	2681	2682	2683				I94-D	WB	235.420	
ML	553	2684	2685	2686				I94-D	WB	234.530	107+90
ML	556	2601	2602	2603				I94-I	EB	233.250	
ML	557	2606	2607	2608				I94-I	EB	233.770	79+70
ML	559	2692	2693	2694				I94-D	WB	233.760	78+10
ML	560	2695	2696	2697	2698			I94-D	WB	233.230	
ML	565	2131	2132					I35W-I	NB	17.020	551+60
ML	566	2135	2136					I35W-I	NB	17.600	82+00
ML	567	2140	2141	2142	2143			I35W-I	NB	18.190	52+00
ML	568	2149	2150	2151				I35W-I	NB	18.930	88+90
ML	569	2153	2154	2155				I35W-I	NB	19.540	120+70
ML	570	2158	2159	2160				I35W-I	NB	19.930	141+30
ML	571	2164	2165	2166				I35W-I	NB	20.350	162+90
ML	572	2167	2168	2169	2170			I35W-I	NB	20.900	194+40
ML	573	2173	2174	2175	2176			I35W-I	NB	21.670	235+30
ML	574	2180	2181	2182				I35W-I	NB	22.250	24+90
ML	575	2184	2185	2186				I35W-I	NB	22.880	51+70
ML	576	2194	2195					I35W-D	SB	22.870	51+20
ML	577	2198	2199	2200				I35W-D	SB	22.240	24+30
ML	578	2202	2203	2204				I35W-D	SB	21.380	220+10
ML	579	2207	2208	2209				I35W-D	SB	20.890	193+90
ML	580	2210	2211	2212				I35W-D	SB	20.400	166+00
ML	581	2215	2216	2217	2218			I35W-D	SB	19.920	140+80
ML	582	2219	2220	2221				I35W-D	SB	19.532	120+30
ML	583	2224	2225	2226				I35W-D	SB	18.760	79+40
ML	584	2229	2230	2231	2232			I35W-D	SB	18.170	111+40
ML	585	2239	2240					I35W-D	SB	17.559	81+70
ML	585	2239	2240					I35W-D	SB	17.590	
ML	586	2242	2243	2244				I35W-D	SB	17.020	51+60
EX	587	2178	2179					I35W-I	NB	22.101	
ML	587	2178	2179					MN36-I	EB	0.001	
ML	590	2250	2251	2252				MN36-I	EB	0.390	66+05
ML	591	2254	2255					MN36-I	EB	0.930	94+80
ML	592	2258	2259					MN36-I	EB	1.210	110+70

ML	593	2264	2265					MN36-I	EB	1.630	1332+70
ML	594	2268	2269					MN36-I	EB	2.380	172+20
ML	595	2271	2272					MN36-I	EB	2.730	190+90
ML	596	2274	2275					MN36-I	EB	3.390	225+50
ML	597	2277	2278					MN36-I	EB	3.870	250+70
ML	598	2280	2281					MN36-I	EB	4.350	275+70
ML	599	2283	2284					MN36-I	EB	4.820	293+30
ML	600	2286	2287					MN36-I	EB	5.190	312+90
ML	601	2290	2291					MN36-I	EB	5.752	344+30
ML	602	2293	2294					MN36-I	EB	6.300	373+60
ML	603	2296	2297					MN36-I	EB	6.700	394+00
ML	605	2313	2314					MN36-D	WB	6.690	393+50
ML	606	2316	2317					MN36-D	WB	6.290	373+10
ML	607	2319	2320					MN36-D	WB	5.660	339+20
ML	608	2323	2324					MN36-D	WB	5.180	312+30
ML	609	2326	2327					MN36-D	WB	4.810	292+70
ML	610	2329	2330					MN36-D	WB	4.180	266+70
ML	611	2332	2333					MN36-D	WB	3.870	250+10
ML	612	2335	2336					MN36-D	WB	3.180	214+10
ML	613	2338	2339					MN36-D	WB	2.730	190+30
ML	614	2341	2342					MN36-D	WB	2.190	162+50
ML	615	2345	2346					MN36-D	WB	1.860	145+30
ML	616	2359	2360					MN36-D	WB	1.200	30
ML	617	2363	2364					MN36-D	WB	0.920	94+30
ML	618	2367	2368					MN36-D	WB	0.380	65+60
ML	619	2370	2371					I35E-I	NB	107.351	
ML	619	2370	2371					I94-I	EB	242.000	
ML	620	2373	2374					I35E-I	NB	107.650	
EX	620	2373	2374					I94-I	EB	242.100	
ML	621	2381	2382	2383	2384			I35E-I	NB	108.110	119+50
ML	622	2386	2387	2388				I35E-I	NB	108.580	384+60
ML	623	2389	2390	2391				I35E-I	NB	108.980	404+60
ML	624	2393	2394	2395				I35E-I	NB	109.420	102+60
ML	625	2397	2398	2399				I35E-I	NB	110.060	136+30
ML	626	2401	2402	2403				I35E-I	NB	110.430	156+00
ML	627	2406	2407	2408				I35E-I	NB	110.940	183+00
ML	628	2410	2411	2412				I35E-I	NB	111.310	202+70
ML	629	2414	2415	2416				I35E-I	NB	111.750	225+70
ML	630	2419	2420	2421				I35E-I	NB	112.650	273+60
ML	633	2426	2427	2428				I35E-D	SB	112.350	258+30
ML	634	2430	2431	2432				I35E-D	SB	111.740	225+30
ML	635	2435	2436	2437				I35E-D	SB	111.300	202+10
ML	636	2442	2443	2444				I35E-D	SB	110.790	175+20
ML	637	2447	2448	2449				I35E-D	SB	110.420	155+60
ML	638	2450	2451	2452				I35E-D	SB	110.060	136+20
ML	639	2458	2459	2460				I35E-D	SB	109.220	92+00
ML	640	2462	2463	2464				I35E-D	SB	108.970	404+10
ML	641	2465	2466	2467				I35E-D	SB	108.570	384+20
ML	642	2469	2470	2471	2472			I35E-D	SB	108.100	119+00
ML	643	2478	2479					I35E-D	SB	107.660	95+00
EN	643	2478	2479					I94-I	EB	242.200	
ML	644	2480	2481					I35E-D	SB	107.120	
EX	644	2480	2481					I94-D	WB	241.935	
ML	652	2701	2702	2703	2704			I35W-I	NB	23.200	70+50
ML	653	2706	2707	2708				I35W-I	NB	23.600	91+90
ML	654	2710	2711	2712				I35W-I	NB	24.190	114+30
ML	655	2714	2715	2716				I35W-I	NB	24.610	146+80
ML	656	2718	2719	2720				I35W-I	NB	24.920	163+30

ML	657	2722	2723	2724				I35W-I	NB	25.470	190+30
ML	658	2726	2727	2728				I35W-I	NB	26.120	223+60
ML	659	897	898	899				I35W-I	NB	26.670	252+10
ML	664	2732	2733	2734				I35W-I	NB	27.140	277+90
ML	665	2736	2737	2738				I35W-I	NB	27.720	308+70
ML	666	2741	2742	2743				I35W-I	NB	28.530	350+30
ML	667	2745	2746	2747				I35W-I	NB	28.810	364+90
ML	668	2750	2751	2752				I35W-I	NB	29.420	397+50
ML	669	2754	2755					I35W-I	NB	29.960	425+80
ML	670	2760	2761					I35W-I	NB	30.770	469+30
ML	671	2764	2765					I35W-I	NB	31.870	527+20
ML	672	2768	2769					I35W-I	NB	33.510	613+60
ML	677	2781	2782					I35W-D	SB	33.510	613+20
ML	678	2785	2786					I35W-D	SB	31.870	526+70
ML	679	2788	2789					I35W-D	SB	30.760	468+80
ML	680	2793	2794					I35W-D	SB	30.040	430+50
ML	681	2797	2798	2799				I35W-D	SB	29.410	397+00
ML	682	2802	2803	2804				I35W-D	SB	28.790	364+30
ML	683	2807	2808	2809				I35W-D	SB	28.200	333+40
ML	684	2812	2813	2814				I35W-D	SB	27.450	294+10
ML	685	2816	2817	2818				I35W-D	SB	27.130	277+30
ML	686	796	797	798				I35W-D	SB	26.660	251+70
ML	691	2822	2823	2824				I35W-D	SB	25.960	215+00
ML	692	2826	2827	2828				I35W-D	SB	25.460	189+80
ML	693	2830	2831	2832				I35W-D	SB	24.920	163+30
ML	694	2834	2835	2836				I35W-D	SB	24.340	132+70
ML	695	2838	2839	2840				I35W-D	SB	24.180	113+80
ML	696	2842	2843	2844				I35W-D	SB	23.590	91+30
ML	697	2846	2847	2848	2849			I35W-D	SB	23.180	69+70
ML	700	2860	2861	2862				I494-D	EB	27.250	882+20
ML	701	2863	2864					I494-D	EB	26.880	862+40
ML	702	2866	2867					I494-D	EB	26.320	832+40
ML	703	2868	2869					I494-D	EB	26.130	817+60
ML	704	2872	2873					I494-D	EB	25.530	791+00
ML	705	2874	2875					I494-D	EB	25.010	764+20
ML	706	2876	2877					I494-D	EB	24.470	735+50
ML	707	2878	2879					I494-D	EB	23.900	705+40
ML	708	2881	2882					I494-D	EB	23.370	676+90
ML	709	2885	2886					I494-D	EB	22.310	620+70
ML	710	2888	2889	2890				I494-D	EB	21.840	596+90
ML	711	2892	2893	2894				I494-D	EB	21.380	575+60
ML	712	2895	2896	2897				I494-D	EB	21.090	557+20
ML	713	2902	2903					I494-D	EB	20.510	527+60
ML	714	2904	2905					I494-D	EB	20.150	508+50
ML	718	2913	2914	2915				I494-I	WB	20.150	508+50
ML	719	2916	2917					I494-I	WB	20.510	528+00
ML	720	2920	2921	2922				I494-I	WB	21.090	557+40
ML	721	2923	2924	2925				I494-I	WB	21.407	576+20
ML	722	2928	2929	2930				I494-I	WB	21.840	597+10
HX	723							I394-I	EB	8.440	
ML	723	2932	2933					I494-I	WB	22.310	620+70
ML	724	2936	2937					I494-I	WB	23.380	678+60
ML	725	2939	2940					I494-I	WB	23.910	705+90
ML	726	2941	2942					I494-I	WB	24.480	736+00
ML	727	2943	2944					I494-I	WB	25.020	764+60
ML	728	2945	2946					I494-I	WB	25.540	791+40
ML	729	2948	2949					I494-I	WB	26.330	832+90
ML	730	2952	2953					I494-I	WB	26.890	862+80

ML	731	2954	2955	2956				I494-I	WB	27.260	882+70
ML	736	2963	2964					US169-D	SB	134.300	599+50
ML	737	2966	2967					US169-D	SB	131.500	457+30
ML	738	2969	2970					US169-D	SB	132.170	487+30
ML	739	2972	2973					US169-D	SB	132.700	515+90
ML	740	2975	2976					US169-D	SB	133.160	539+50
ML	741	2978	2979					US169-D	SB	133.560	560+70
ML	742	2983	2984					US169-D	SB	134.580	614+70
ML	743	2988	2989					US169-D	SB	135.090	639+30
ML	744	2991	2992					US169-D	SB	135.470	671+40
ML	745	2994	2995					US169-D	SB	136.030	690+90
ML	746	2998	2999					US169-D	SB	136.570	719+30
ML	747	3001	3002					US169-D	SB	136.970	740+30
ML	748	3004	3005					US169-D	SB	137.280	757+00
ML	749	3007	3008					US169-D	SB	137.990	794+00
ML	750	3010	3011					US169-D	SB	138.850	
ML	751	3012	3013					US169-D	SB	138.510	
ML	755	3018	3019					US169-I	NB	134.580	614+70
ML	756	3021	3022					US169-I	NB	138.510	
ML	757	3023	3024					US169-I	NB	138.850	
ML	758	3026	3027					US169-I	NB	137.990	794+00
ML	759	3028	3029					US169-I	NB	137.280	757+00
ML	760	3031	3032					US169-I	NB	136.850	734+20
ML	761	3034	3035					US169-I	NB	136.570	719+30
ML	762	3037	3038					US169-I	NB	136.030	690+90
ML	763	3041	3042					US169-I	NB	135.470	671+40
ML	764	3044	3045					US169-I	NB	135.240	649+10
ML	765	3048	3049					US169-I	NB	134.300	599+50
ML	766	3053	3054					US169-I	NB	133.710	569+00
ML	767	3056	3057					US169-I	NB	133.160	539+50
ML	768	3059	3060					US169-I	NB	132.700	515+90
ML	769	3062	3063					US169-I	NB	132.170	487+30
EN	770	3065	3066					US169-I	NB	131.680	461+30
HR	2083							I394-I	EB	5.800	
CD	2538	2235	35W/4STSM					I35W-D	SB	18.110	
EX		2285	35E/36SX					I35E-D	SB	111.610	
EX		2325						I35E-D	SB	111.826	
EX		2425	35E/LITCSX					I35E-D	SB	112.698	
EN		2429	35E/LITCSM					I35E-D	SB	112.340	
EN		2433	35E/36SM1					I35E-D	SB	111.710	
EN		2434	35E/36SM2					I35E-D	SB	111.502	
EX		2438	35E/ROSESX					I35E-D	SB	111.006	
EN		2445	35E/ROSESM					I35E-D	SB	110.695	
EX		2446	35E/LARPSX					I35E-D	SB	110.528	
EN		2456	35E/WHELSM					I35E-D	SB	110.018	
EX		2457	35E/MARYSX					I35E-D	SB	109.508	
EN		2461	35E/MARYSM					I35E-D	SB	109.148	
EX		2468	35E/PENNSX					I35E-D	SB	108.314	
EX		2473	35E/UNIVSX					I35E-D	SB	108.153	
EX		2474	35E/94SX					I35E-D	SB	107.925	
CD		2475						I35E-D	SB	107.640	
CD		2476	35E/TH3SX					I35E-D	SB	107.630	
EN		2477						I35E-D	SB	107.900	
EN		2482	35E/WABSM					I35E-D	SB	107.050	
EX		2288	35E/36NX					I35E-I	NB	111.470	
EX		2322						I35E-I	NB	111.715	
EX		2357	35E/PETENX					I35E-I	NB	106.901	
EX		2372						I35E-I	NB	107.050	

HN		2378	35E/BRDNMH					I35E-I	NB	108.070	
EX		2380	35E/PENNNX					I35E-I	NB	108.088	
EN		2385	35E/PENNNM					I35E-I	NB	108.306	
EX		2392	35E/MARYNX					I35E-I	NB	109.187	
EN		2396	35E/MARYNM					I35E-I	NB	109.525	
EX		2400	35E/WHELNX					I35E-I	NB	110.111	
EN		2404	35E/LARPNM					I35E-I	NB	110.559	
EX		2405	35E/ROSENX					I35E-I	NB	110.722	
EN		2409	35E/ROSENM					I35E-I	NB	111.042	
EN		2413	35E/36NM1					I35E-I	NB	111.603	
EN		2417	35E/36NM2					I35E-I	NB	111.832	
EX		2418	35E/LITCNX					I35E-I	NB	112.342	
EN		2422	35E/LITCNM					I35E-I	NB	112.659	
HN		7	35W/98THSMH					I35W-D	SB	6.046	
EX		11	35W/TH13SX					I35W-D	SB	2.606	
EN		12	35W/TH13SM					I35W-D	SB	2.506	
EN		96	35W/31STSM					I35W-D	SB	15.309	
EN		97	35W/36THSM					I35W-D	SB	14.638	
EN		98	35W/46THSM					I35W-D	SB	13.388	
EN		99	35W/DIAMSM					I35W-D	SB	12.243	
EN		100	35W/WB62SM					I35W-D	SB	11.520	
EN		103	35W/66THSM					I35W-D	SB	10.094	
EN		104	35W/W494SM					I35W-D	SB	8.795	
EN		105	35W/E494SM					I35W-D	SB	8.599	
EN		106	35W/82NDSM					I35W-D	SB	8.182	
EN		107	35W/90THSM					I35W-D	SB	7.185	
EN		108	35W/94THSM					I35W-D	SB	6.581	
EN		110	35W/106THSM					I35W-D	SB	5.113	
EX		135	35W/35THSX					I35W-D	SB	15.192	
EX		136	35W/46THSX					I35W-D	SB	13.868	
EX		137	35W/DIAMSX					I35W-D	SB	12.668	
EX		138	35W/60THSX					I35W-D	SB	11.987	
EX		139	35W/E62SX					I35W-D	SB	11.726	
EX		140	35W/LYNDSX					I35W-D	SB	11.134	
EX		143	35W/66THSX					I35W-D	SB	10.425	
EX		144	35W/76THSX					I35W-D	SB	9.141	
EX		147	35W/82NDSX					I35W-D	SB	8.387	
EX		148	35W/90THSX					I35W-D	SB	7.400	
EX		149	35W/94THSX					I35W-D	SB	6.840	
EX		150	35W/98THSX					I35W-D	SB	6.350	
EX		151	35W/106THSX					I35W-D	SB	5.340	
EX		486						I35W-D	SB	26.905	
EN		737	35W/694SM1					I35W-D	SB	26.792	
EX		738						I35W-D	SB	26.673	
EN		799	35W/694SM2					I35W-D	SB	26.565	
EX		862						I35W-D	SB	8.681	
EX		867						I35W-D	SB	8.883	
EX		1004	35W/113THSX					I35W-D	SB	4.198	
EN		1005	35W/113THSM					I35W-D	SB	4.109	
EX		1009	35W/CLIFFSX					I35W-D	SB	3.414	
EN		1010	35W/CLIFFSM					I35W-D	SB	3.270	
EX		1017	35W/131STSX					I35W-D	SB	2.262	
EN		1032	35W/131STSM					I35W-D	SB	2.071	
EN		1072	35W/121SM					I35W-D	SB	10.600	
EX		2196	35W/280SX1					I35W-D	SB	22.900	
EX		2197	35W/280SX2					I35W-D	SB	22.292	
EX		2201	35W/INDSSX					I35W-D	SB	21.667	
EN		2205	35W/INDSSM					I35W-D	SB	21.330	

EX		2206	35W/STINSX					I35W-D	SB	20.961	
EN		2213	35W/TH88SM					I35W-D	SB	20.387	
EN		2214	35W/JOHNSM					I35W-D	SB	20.243	
EN		2222	35W/HENNSM					I35W-D	SB	19.373	
EX		2223	35W/UNIVSX					I35W-D	SB	19.007	
EN		2227	35W/UNIVSM					I35W-D	SB	18.732	
EX		2228	35W/WASHSX					I35W-D	SB	18.229	
CD		2233	35W/WASHSM					I35W-D	SB	17.790	
CD		2234	35W/TH122SM					I35W-D	SB	18.100	
CD		2236	35W/55EX1					I35W-D	SB	18.000	
EN		2241	35W/CDSM					I35W-D	SB	17.475	
EN		2245	35W/TH122SMH					I35W-D	SB		
HR		2246	35W/CD001	35w				I35W-D	SB		
CD		2689	35W/CDSX					I35W-D	SB	17.500	
EX		2780	35W/LEXSX					I35W-D	SB	33.752	
EN		2783	35W/LEXSM					I35W-D	SB	33.449	
EX		2784	35W/CR10SX					I35W-D	SB	32.114	
EN		2787	35W/CR10SM					I35W-D	SB	31.807	
EX		2792	35W/TH118SX					I35W-D	SB	30.302	
EN		2795	35W/TH118SM					I35W-D	SB	30.001	
EX		2796	35W/CRISX					I35W-D	SB	29.512	
EN		2800	35W/CRISM					I35W-D	SB	29.338	
EX		2801	35W/CRHSX					I35W-D	SB	28.893	
EN		2805	35W/CRHSM					I35W-D	SB	28.752	
EX		2806	35W/TH10SX					I35W-D	SB	28.483	
EN		2810	35W/TH10SM					I35W-D	SB	28.160	
EX		2811	35W/TH96SX					I35W-D	SB	27.740	
EN		2815	35W/TH96SM					I35W-D	SB	27.388	
EX		2821	35W/CRE2SX					I35W-D	SB	26.214	
EN		2825	35W/CRE2SM					I35W-D	SB	25.897	
EX		2829	35W/TH88SX					I35W-D	SB	25.020	
EX		2833	35W/CRDSX					I35W-D	SB	24.429	
EN		2837	35W/CRDSM					I35W-D	SB	24.298	
EX		2841	35W/CRCSX					I35W-D	SB	23.727	
EN		2845	35W/CRCSM					I35W-D	SB	23.550	
HN		2	35W/CLIFFNMH					I35W-I	NB	3.384	
HN		3	35W/98THNMH					I35W-I	NB	6.360	
HN		4	35W/76THNMH					I35W-I	NB	9.093	
HN		5	35W/TH13NMH					I35W-I	NB	2.876	
HN		6	35W/66THNMH					I35W-I	NB	10.534	
HN		13	35W/CR42NMH					I35W-I	NB	0.764	
EN		111	35W/131STNM					I35W-I	NB	2.244	
EN		112	35W/TH13NM1					I35W-I	NB	2.637	
EN		115	35W/106THNM					I35W-I	NB	5.385	
EN		117	35W/94THNM					I35W-I	NB	6.838	
EN		118	35W/90THNM					I35W-I	NB	7.426	
EN		119	35W/82NDNM					I35W-I	NB	8.415	
EN		120	35W/E494NM					I35W-I	NB	8.678	
EN		121	35W/W494NM					I35W-I	NB	8.903	
EN		123	35W/66THNM					I35W-I	NB	10.535	
EN		125	35W/LYNDNM					I35W-I	NB	11.163	
EN		126	35W/W62NM					I35W-I	NB	11.716	
EN		127	35W/60THNM					I35W-I	NB	11.981	
EN		128	35W/DIAMNM					I35W-I	NB	12.642	
EN		129	35W/46THNM					I35W-I	NB	13.842	
EN		130	35W/35THNM					I35W-I	NB	15.171	
EX		152	35W/131STNX					I35W-I	NB	2.049	
EX		153	35W/TH13NX1					I35W-I	NB	2.486	

EX		154	35W/TH13NX2					I35W-I	NB	2.730	
EX		155	35W/CLIFFNX					I35W-I	NB	3.251	
EX		156	35W/106THNX					I35W-I	NB	5.112	
EX		157	35W/98THNX					I35W-I	NB	6.059	
EX		158	35W/94THNX					I35W-I	NB	6.579	
EX		159	35W/90THNX					I35W-I	NB	7.195	
EX		160	35W/82NDNX					I35W-I	NB	8.163	
EX		162						I35W-I	NB	8.792	
EX		163	35W/66THNX					I35W-I	NB	10.141	
EX		167	35W/E62NX					I35W-I	NB	11.535	
EX		168	35W/DAIMNX					I35W-I	NB	12.201	
EX		169	35W/46THNX					I35W-I	NB	13.424	
EX		170	35W/36THNX					I35W-I	NB	14.651	
EX		171	35W/31STNX					I35W-I	NB	15.312	
EN		386	35W/CR42NM					I35W-I	NB	0.764	
EX		492	35W/113THNX					I35W-I	NB	3.999	
EN		493	35W/113THNM					I35W-I	NB	4.115	
EN		800	35W/694NM2					I35W-I	NB	26.935	
EX		861						I35W-I	NB	26.814	
EX		863						I35W-I	NB	8.602	
EX		866						I35W-I	NB	26.583	
EN		975	35W/694NM1					I35W-I	NB	26.678	
EN		2130	35W/5AVNM					I35W-I	NB	16.991	
EX		2139	35W/3STNX					I35W-I	NB	17.900	
EN		2144	35W/WASHNM					I35W-I	NB	18.200	
EN		2152	35W/UNIVNM					I35W-I	NB	18.974	
EX		2156	35W/BRDNX1					I35W-I	NB	19.600	
EX		2157	35W/BRDNX2					I35W-I	NB	19.938	
EX		2163	35W/JOHNNX					I35W-I	NB	20.300	
EN		2171	35W/STINNM					I35W-I	NB	20.944	
EX		2172	35W/INDSNX					I35W-I	NB	21.374	
EN		2177	35W/INDSNM					I35W-I	NB	21.727	
EN		2180						I35W-I	NB	22.251	
CD		2187	35W/CLVNM					I35W-I	NB	23.000	
EN		2188	35W/TH36NM					I35W-I	NB	23.059	
EN		2191	35W/94NM					I35W-I	NB	17.082	
EX		2609						I35W-I	NB	17.388	
EX		2705	35W/CRCNX					I35W-I	NB	23.539	
EN		2709	35W/CRCNM					I35W-I	NB	23.716	
EX		2713	35W/CRDNX					I35W-I	NB	24.376	
EN		2717	35W/CRDNM					I35W-I	NB	24.687	
EN		2721	35W/TH88NM					I35W-I	NB	25.120	
EX		2725	35W/CRE2NX					I35W-I	NB	25.897	
EN		2729	35W/CRE2NM					I35W-I	NB	26.213	
EX		2735	35W/TH96NX					I35W-I	NB	27.367	
EN		2739	35W/TH96NM					I35W-I	NB	27.766	
EX		2740	35W/TH10NX					I35W-I	NB	28.201	
EN		2744	35W/TH10NM					I35W-I	NB	28.531	
EN		2748	35W/CRHNM					I35W-I	NB	28.820	
EX		2749	35W/CRINX					I35W-I	NB	29.330	
EN		2753	35W/CRINM					I35W-I	NB	29.521	
EN		2758	35W/TH118NM					I35W-I	NB	29.976	
EX		2759	35W/CRJNX					I35W-I	NB	30.537	
EX		2762	35W/LKDRNX					I35W-I	NB	30.760	
EX		2763	35W/CR10NX					I35W-I	NB	31.828	
EN		2766	35W/CR10NM					I35W-I	NB	32.156	
EX		2767	35W/LEXNX					I35W-I	NB	33.400	
EN		2770	35W/LEXNM					I35W-I	NB	33.703	

HN		559	394/10STWH					I394-D	WB	9.118	
HN		560	394/HATHWH					I394-D	WB	8.844	
CD		675	394/E94WM					I394-D	WB	8.555	
CD		676	394/LYNDWM					I394-D	WB	8.551	
HN		707	394/WMHOV					I394-D	WB	8.660	
HN		708	394/W94WMH					I394-D	WB	8.500	
HN		763	394/DUNWMH					I394-D	WB	8.300	
EN		766	394/DUNWM					I394-D	WB	8.239	
EN		787	394/PENWM					I394-D	WB	7.386	
HR		1714	394/W100H					I394-D	WB	5.700	
HR		1716						I394-D	WB	5.800	
HR		1717	394/FNTGWXH					I394-D	WB	5.800	
CD		1743	394/100WX1					I394-D	WB	6.100	
HR		1748	394/100WXH					I394-D	WB	6.050	
CD		1749	394/100WM1					I394-D	WB	5.975	
CD		1752	394/100CW3					I394-D	WB	5.760	
CD		1753	394/100WM2					I394-D	WB	5.728	
CD		1759	394/XENWMH					I394-D	WB	5.195	
EX		1764	394/LOUSWX					I394-D	WB	4.655	
HN		1768	394/LOUWMH					I394-D	WB	4.305	
EX		1774	394/169CDWX					I394-D	WB	3.585	
CD		1775	394/GENWX					I394-D	WB	3.400	
CD		1779	394/169CDW					I394-D	WB	3.100	
CD		1783	394/169WM1					I394-D	WB	3.002	
CD		1786	394/S169WM					I394-D	WB	2.746	
EX		1791	394/CR73WX					I394-D	WB	2.150	
EN		1795	394/CR73WM					I394-D	WB	1.987	
EX		1796	394/RIDGWX					I394-D	WB	1.511	
EX		1800	394/PLYWX					I394-D	WB	0.915	
EN		1804	394/PLYWM					I394-D	WB	0.763	
EX		1805	394/494WX1					I394-D	WB	0.327	
CD		1809	394/494WX2					I394-D	WB	0.190	
CD		1810	394/494WM					I394-D	WB	0.047	
CD		1863						I394-D	WB	0.195	
CD		1938						I394-D	WB	3.120	
CD		1966						I394-D	WB	2.913	
CD		2088	394/100WX2					I394-D	WB	5.872	
EX		646	394/12STEX					I394-I	EB	8.838	
HN		670	394/EXHOV					I394-I	EB	8.650	
HX		723						I394-I	EB	8.432	
HX		764						I394-I	EB	8.300	
EX		765	394/DUNEX					I394-I	EB	8.249	
EN		782	394/PENEM					I394-I	EB	7.648	
EX		786	394/PENEX					I394-I	EB	7.358	
CD		1649	394/494EMR					I394-I	EB	0.198	
CD		1650	394/494EML					I394-I	EB	0.180	
HO		1664	394/RDGEMH					I394-I	EB	1.490	
EN		1667	394/RDGEM					I394-I	EB	1.492	
EX		1668	394/CR73EX					I394-I	EB	1.971	
HO		1672	394/73EMH					I394-I	EB	2.110	
EN		1675	394/73EM					I394-I	EB	2.119	
CD		1680						I394-I	EB	2.900	
CD		1681						I394-I	EB	3.160	
CD		1685	394/GENEX					I394-I	EB	3.457	
HO		1691						I394-I	EB	3.550	
EX		1697	394/LOUSEX					I394-I	EB	4.301	
CD		1706	394/100CDX					I394-I	EB	5.202	
CD		1707	394/XENIEX					I394-I	EB	5.202	

CD		1711	394/XENIEM					I394-I	EB	5.545
CD		1715	394/100CDE					I394-I	EB	5.720
HR		1727	394/CL100EMH					I394-I	EB	6.000
HN		1735	394/LOUEMH					I394-I	EB	4.606
CD		1860						I394-I	EB	0.048
CD		1935						I394-I	EB	2.988
CD		1969						I394-I	EB	2.784
CD		2035						I394-I	EB	5.995
HR		2089						I394-I	EB	5.717
HR		555	394/3AVBUS					I394-R	EB/WB	9.156
HR		773	394/DUNH1					I394-R	EB/WB	7.700
HR		774	394/DUNH2					I394-R	EB/WB	7.700
HR		780	394/PENH1					I394-R	EB/WB	7.560
HR		781	394/PENH2					I394-R	EB/WB	7.560
HR		791	394/WIRH2					I394-R	EB/WB	7.000
HR		792	394/WIRH2					I394-R	EB/WB	7.000
CD		1718	394/S100EM					I394-R	EB/WB	5.855
EN		1724	394/100CDEM					I394-R	EB/WB	6.070
HR		1725	394/CEDREMH					I394-R	EB/WB	6.000
CD		1726	394/N100EMH					I394-R	EB/WB	6.000
HR		1733	394/WIRTH1					I394-R	EB/WB	6.560
HR		1734	394/WIRTH2					I394-R	EB/WB	6.560
HR		1746	394/E100HN					I394-R	EB/WB	6.160
HR		1747	394/E100HS					I394-R	EB/WB	6.160
Q		3082	494/N100EQ1	494				I-494	EB	T.H.100 NB
Q		3083	494/N100EQ2	494				I-494	EB	T.H.100 NB
Q		3084	494/SFRANEQ2	494				I-494	EB	France SB
Q		3085	494/SFRANEQ3	494				I-494	EB	France SB
Q		3086	494/NFRANEQ1	494				I-494	EB	France NB
Q		3087	494/NFRANEQ2	494				I-494	EB	France NB
Q		3088	494/PENNEQ	494				I-494	EB	Penn Ave
Q		3089	494/S35WEQ	494				I-494	EB	I-35W SB
Q		3090	494/N35WEQ1	494				I-494	EB	I-35W NB
Q		3091	494/N35WEQ2	494				I-494	EB	I-35W NB
Q		3092	494/LYNDEQ	494				I-494	EB	Lyndale Ave
Q		3093	494/NICEQ	494				I-494	EB	Nicollet Ave
Q		3094	494/I2AVEQ	494				I-494	EB	12th Ave
Q		3095	494/SFRANEQ1	494				I-494	EB	France SB
Q		3096	494/SFRANEQ4	494				I-494	EB	France SB
Q		3097	494/SFRANEQ5	494				I-494	EB	France SB
Q		3098	494/SFRANEQ6	494				I-494	EB	France SB
Q		3101	494/34AVWQ1	494				I-494	WB	34th Ave
Q		3102	494/34AVWQ2	494				I-494	WB	34th Ave
Q		3103	494/24AVWQ1	494				I-494	WB	24th Ave
Q		3104	494/24AVWQ2	494				I-494	WB	24th Ave
Q		3108	494/STH77WQ1	494				I-494	WB	T.H.77 SB
Q		3109	494/STH77WQ2	494				I-494	WB	T.H.77 SB
Q		3110	494/PORTWQ	494				I-494	WB	Portland Ave
Q		3111	494/NICWQ	494				I-494	WB	Nicollet Ave
Q		3112	494/LYNDWQ	494				I-494	WB	Lyndale Ave
Q		3113	494/S35WWQ1	494				I-494	WB	I-35W SB
Q		3114	494/S35WWQ2	494				I-494	WB	I-35W SB
Q		3115	494/PENNWQ	494				I-494	WB	Penn Ave
Q		3116	494/NFRANWQ	494				I-494	WB	France NB
Q		3117	494/SFRANWQ1	494				I-494	WB	France SB
Q		3118	494/SFRANWQ2	494				I-494	WB	France SB
Q		3119	494/N100WQ	494				I-494	WB	T.H.100 NB
Q		3120	494/S100WQ	494				I-494	WB	T.H.100 SB

EX		105						I494-D	EB	5.488	
EX		120						I494-D	EB	5.283	
EN		805	494/EBSHEM					I494-D	EB	8.385	
EN		822	494/S100EM					I494-D	EB	7.976	
EN		823	494/N100EM					I494-D	EB	7.767	
EX		824	494/S100EX					I494-D	EB	8.081	
EX		825	494/N100EX					I494-D	EB	7.868	
EN		840	494/SFRAEM					I494-D	EB	6.950	
EN		841	494/NFRAEM					I494-D	EB	6.660	
EX		842	494/FRANEX					I494-D	EB	7.039	
EN		852	494/PENNEM					I494-D	EB	5.721	
EX		853	494/PENNEX					I494-D	EB	6.016	
EN		862	494/S35WEM					I494-D	EB	5.390	
EN		863	494/N35WEM					I494-D	EB	5.190	
EN		876	494/LYNDEM					I494-D	EB	4.736	
EX		877	494/LYNDEX					I494-D	EB	4.976	
EN		886	494/NICEM					I494-D	EB	4.205	
EX		887	494/NICEX					I494-D	EB	4.522	
EX		895	494/PORTEX					I494-D	EB	3.943	
EN		997	494/12AVEM					I494-D	EB	3.220	
EX		1182	494/62SX					I494-D	EB	14.037	
EN		1185	494/62SM					I494-D	EB	13.734	
EX		1191	494/VVRSX					I494-D	EB	12.484	
EX		1192	494/5SX1					I494-D	EB	12.148	
EN		1193	494/5SM1					I494-D	EB	12.012	
EX		1194	494/5SX2					I494-D	EB	11.874	
EN		1197	494/5SM2					I494-D	EB	11.711	
EN		1200	494/EDENEM					I494-D	EB	10.946	
EX		1201	494/169EX					I494-D	EB	10.346	
EN		1204	494/169EM					I494-D	EB	9.968	
EN		1529						I494-D	EB	2.878	
EX		1530	494/TH77EX					I494-D	EB	2.773	
EX		1531	494/24AVEX					I494-D	EB	2.559	
EN		1535	494/NTH77EM					I494-D	EB	2.255	
EN		1536	494/24AVEM					I494-D	EB	2.085	
CD		1543	494/34AVEM					I494-D	EB	1.500	
CD		1544						I494-D	EB	1.510	
EX		1648						I494-D	EB	19.305	
EX		1815						I494-D	EB	19.561	
EN		1868	494/W394SM					I494-D	EB	19.525	
EN		1871	494/E394SM					I494-D	EB	19.292	
EX		1876	494/MTKASX					I494-D	EB	17.621	
EN		1879	494/MTKASM					I494-D	EB	17.466	
EX		1882	494/TH7SX1					I494-D	EB	16.483	
EN		1885	494/TH7SM1					I494-D	EB	16.210	
EX		1886	494/TH7SX2					I494-D	EB	16.198	
EN		1887	494/TH7SM2					I494-D	EB	16.083	
EX		2865	494/BASLKSX					I494-D	EB	26.389	
HN		2871	494/BASLKSMH					I494-D	EB	26.029	
EX		2880						I494-D	EB	23.677	
EN		2883						I494-D	EB	23.345	
EX		2884						I494-D	EB	22.469	
EN		2887						I494-D	EB	22.091	
EX		2891						I494-D	EB	21.442	
EN		2898						I494-D	EB	21.082	
EX		2901						I494-D	EB	20.645	
EN		2906						I494-D	EB	20.086	
EX		104						I494-I	WB	5.406	

EX		121						I494-I	WB	5.187	
EN		162						I494-I	WB	5.293	
EN		806	494/S100WM					I494-I	WB	8.101	
EX		807	494/S100WX					I494-I	WB	7.990	
EX		814	494/EBSHWX					I494-I	WB	8.351	
EN		817	494/N100WM					I494-I	WB	7.883	
EX		818	494/N100WX					I494-I	WB	7.762	
EN		829	494/NFRAWM					I494-I	WB	6.810	
EX		830	494/FRANWX					I494-I	WB	6.650	
EN		857	494/PENNWM					I494-I	WB	6.039	
EX		858	494/PENNWX					I494-I	WB	5.757	
EN		867	494/S35WWM					I494-I	WB	5.513	
EN		871	494/LYNDWM					I494-I	WB	4.988	
EX		872	494/LYNDWX					I494-I	WB	4.743	
EN		881	494/NICWM					I494-I	WB	4.517	
EX		882	494/NICWX					I494-I	WB	4.219	
EN		894	494/PORTWM					I494-I	WB	3.963	
EN		896	494/SFRAWM					I494-I	WB	7.046	
EX		998	494/12AVWX					I494-I	WB	3.176	
EX		1213	494/169WX					I494-I	WB	9.930	
EN		1216	494/169WM					I494-I	WB	10.278	
EX		1221	494/5NX1					I494-I	WB	11.700	
EN		1222	494/5NM1					I494-I	WB	11.817	
EX		1223	494/5NX2					I494-I	WB	11.980	
EN		1226	494/5NM2					I494-I	WB	12.188	
EN		1227	494/VVRNM					I494-I	WB	12.410	
EX		1233	494/62NX					I494-I	WB	13.733	
EN		1236	494/62NM					I494-I	WB	14.075	
EX		1487						I494-I	WB	2.879	
EN		1602	494/34AVWM					I494-I	WB	1.782	
EX		1608	494/24AVWX					I494-I	WB	2.047	
EX		1613	494/NTH77WX					I494-I	WB	2.216	
HN		1614	494/24AVWMH					I494-I	WB	2.562	
HN		1618	494/TH77WMH					I494-I	WB	2.772	
EN		1649						I494-I	WB	19.303	
EX		1810						I494-I	WB	19.536	
EX		1842	494/TH7NX1					I494-I	WB	16.053	
EN		1845	494/TH7NM1					I494-I	WB	16.226	
EX		1846	494/TH7NX2					I494-I	WB	16.330	
EN		1847	494/TH7NM2					I494-I	WB	16.476	
EX		1850	494/MTKANX					I494-I	WB	17.469	
EN		1853	494/MTKANM					I494-I	WB	17.614	
EN		1860	494/394NM1					I494-I	WB	19.314	
EN		1863	494/394NM2					I494-I	WB	19.585	
EX		2912						I494-I	WB	20.141	
EN		2918						I494-I	WB	20.530	
EX		2919						I494-I	WB	21.087	
EN		2927						I494-I	WB	21.441	
EX		2931						I494-I	WB	22.049	
EN		2934						I494-I	WB	22.450	
EX		2935						I494-I	WB	23.316	
EN		2938						I494-I	WB	23.644	
EX		2947	494/BASLKNX					I494-I	WB	26.012	
HN		2951	494/BASLKNMH					I494-I	WB	26.340	
EN		20	694/LLWM					I694-D	WB	39.827	
EX		447	694/E94WX					I694-D	WB	35.769	
EX		449	694/LLWX					I694-D	WB	40.110	
EX		450	694/252WX					I694-D	WB	35.933	

EN		486	694/35WWM2					I694-D	WB	40.571	
EX		554	694/65WX1					I694-D	WB	37.950	
EN		557	694/NB65WM					I694-D	WB	37.759	
EX		558	694/SB65WX					I694-D	WB	37.651	
EN		561	694/SB65WM					I694-D	WB	37.494	
EX		562	694/TH47WX					I694-D	WB	37.077	
EN		565	694/TH47WM					I694-D	WB	36.758	
EX		566	694/ERIVWX					I694-D	WB	36.422	
EN		568	694/ERIVWM					I694-D	WB	36.259	
EX		645	694/100WX					I694-D	WB	35.552	
EN		732	694/ERRWM1					I694-D	WB		
EN		733	696/ERRWM2					I694-D	WB		
EX		737						I694-D	WB	40.721	
EN		753	694/SILWM					I694-D	WB	39.018	
EX		754	694/SILWX					I694-D	WB	39.329	
EX		800						I694-D	WB	40.964	
EN		861	694/35WWM1					I694-D	WB	40.839	
EX		421	694/65WX2					I694-I	EB		
EX		448	694/LLEX					I694-I	EB	39.853	
EN		485	694/LLEM					I694-I	EB	40.118	
EX		574	694/252EX					I694-I	EB	35.885	
EN		575	694/252EM					I694-I	EB	35.774	
EN		576	694/W94EM					I694-I	EB	35.968	
EX		578	694/ERIVEX					I694-I	EB	36.225	
EN		580	694/ERIVEM					I694-I	EB	36.396	
EX		581	694/TH47EX					I694-I	EB	36.756	
EN		584	694/TH47EM					I694-I	EB	37.108	
EX		585	694/S65EX					I694-I	EB	37.494	
EN		586	694/S65EM					I694-I	EB	37.672	
EX		590	694/N65EX					I694-I	EB	37.771	
EN		592	694/N65EM					I694-I	EB	37.953	
EN		671	694/SHINEM					I694-I	EB	35.169	
EN		692	694/E94EM					I694-I	EB	35.660	
EN		738	694/S35WEM					I694-I	EB	40.712	
EX		755	694/SILEX					I694-I	EB	39.031	
EN		756	694/SILEM					I694-I	EB	39.335	
EX		799						I694-I	EB	40.575	
EN		866	694/35WEM2					I694-I	EB	40.955	
EX		975						I694-I	EB	40.833	
CD		31	94/7THWM					I94-D	WB	230.100	
EN		48	94/BROADWM					I94-D	WB	229.183	
EX		71	94/DOWLWX					I94-D	WB	228.184	
EN		77	94/DOWLWM					I94-D	WB	227.751	
EX		175	94/11THWX					I94-D	WB	233.118	
EN		455	94/53WM					I94-D	WB	225.757	
EX		610	94/49WX					I94-D	WB	226.732	
EX		612	94/394WXL					I94-D	WB	231.439	
EX		613	94/394WXR					I94-D	WB	231.439	
EX		620	94/OLSONWX					I94-D	WB	231.189	
EN		648	94/252WM					I94-D	WB	224.142	
EX		656	94/CR152WX					I94-D	WB	222.603	
EN		739	94/SHINWM					I94-D	WB	223.481	
EX		740	94/SHINWX					I94-D	WB	223.746	
EN		931	94/CR61WM					I94-D	WB	217.480	
EX		932	94/CR61WX					I94-D	WB	217.738	
EN		943	94/S169WM					I94-D	WB	218.761	
EX		944	94/S169WX					I94-D	WB	218.904	
EX		947	94/N169WX					I94-D	WB	219.169	

EN		948	94/N169WM					I94-D	WB	219.017	
EN		956	94/BOONWM					I94-D	WB	219.520	
EX		957	94/BOONWX					I94-D	WB	219.747	
EN		968	94/CR81WM					I94-D	WB	220.488	
EX		969	94/CR81WX					I94-D	WB	220.657	
EN		982	94/152WM					I94-D	WB	222.324	
EN		1890	94/WABWM					I94-D	WB	241.848	
EX		1896	94/12STWX					I94-D	WB	242.300	
EX		2377						I94-D	WB	242.378	
EN		2477	94/UNIVWM					I94-D	WB	242.052	
HN		2657	94/MARIWMH					I94-D	WB	241.056	
EN		2658	94/DALEWM					I94-D	WB	240.156	
EN		2667	94/LEXWM					I94-D	WB	239.303	
EN		2671	94/SNELWM					I94-D	WB	238.230	
EN		2679	94/TH280WM					I94-D	WB	236.213	
EN		2680	94/HURONWM					I94-D	WB	235.332	
EN		2687	94/25AVEWM					I94-D	WB	234.516	
EX		2688	94/5STWX					I94-D	WB	234.023	
EN		2690	94/CDWM					I94-D	WB	233.511	
EN		2691	94/HIAWWM					I94-D	WB	233.511	
EX		2699						I94-D	WB	241.426	
HN		2700	94/VANDWMH					I94-D	WB	236.896	
EX		37	94/BROADEX					I94-I	EB	229.204	
EN		64	94/DOWLEM					I94-I	EB	228.152	
EX		95						I94-I	EB	232.882	
HN		420	94/DOWLEH					I94-I	EB	228.152	
EX		511	94/53EX					I94-I	EB	225.731	
EN		609	94/47EM					I94-I	EB	226.767	
HN		660	94/CR152EH					I94-I	EB	222.646	
EX		669	94/SHINEX					I94-I	EB	223.822	
EN		672	94/SHINEM					I94-I	EB	224.137	
EX		692						I94-I	EB	224.130	
CD		693						I94-I	EB	226.601	
CD		694						I94-I	EB	226.600	
CD		695						I94-I	EB	226.500	
EX		700	94/DOWLEX					I94-I	EB	227.678	
HN		723	94/394EXH					I94-I	EB	231.615	
EN		906	94/WEAVEM					I94-I	EB	215.700	
EN		930	94/CR61EM					I94-I	EB	217.768	
EX		933	94/CR61EX					I94-I	EB	217.558	
EN		942	94/S169EM					I94-I	EB	218.904	
EX		945	94/S169EX					I94-I	EB	218.658	
EX		946	94/N169EX					I94-I	EB	219.017	
EN		949	94/N169EM					I94-I	EB	219.169	
EX		955	94/BOONEX					I94-I	EB	219.520	
EN		958	94/BOONEM					I94-I	EB	219.747	
EX		967	94/CR81EX					I94-I	EB	220.541	
EN		970	94/CR81EM					I94-I	EB	220.740	
EX		981	94/152EX					I94-I	EB	222.347	
EN		1894	94/JACKEM					I94-I	EB	242.243	
EX		2191						I94-I	EB	233.085	
EN		2475	94/S35EEM					I94-I	EB	242.351	
HN		2500	94/SNELWMH					I94-I	EB		
HN		2600	94/6STEMH					I94-I	EB		
EN		2604						I94-I	EB	233.280	
EX		2605						I94-I	EB	233.693	
EN		2609						I94-I	EB	233.695	
EN		2610						I94-I	EB	234.105	

EN		2615	94/CEDAREM					I94-I	EB	234.209	
EN		2616	94/RVSDEM					I94-I	EB	234.991	
EN		2624	94/HURONEM					I94-I	EB	235.517	
EX		2628	94/TH280EX					I94-I	EB	236.462	
EN		2629	94/CRETEM					I94-I	EB	237.292	
EN		2637	94/SNELEM					I94-I	EB	238.723	
EN		2642	94/LEXEM					I94-I	EB	239.634	
EN		2647	94/DALEEM					I94-I	EB	240.615	
EX		2651	94/10STEX					I94-I	EB	241.214	
EN		2652	94/MARIEM					I94-I	EB	241.540	
EX		2190	36/CLEVWX					MN36-D	WB	0.242	
EX		2310	36/61WX1					MN36-D	WB	6.956	
EN		2311	36/61WM1					MN36-D	WB	6.898	
EX		2312	36/61WX2					MN36-D	WB	6.787	
EN		2315	36/61WM2					MN36-D	WB	6.634	
EX		2318	36/EDGERWX					MN36-D	WB	5.927	
EN		2321	36/EDGERWM					MN36-D	WB	5.628	
EN		2322	36/35EWM1					MN36-D	WB	5.160	
EN		2325	36/35EWM2					MN36-D	WB	4.944	
EX		2328	36/RICEWX					MN36-D	WB	4.416	
EN		2331	36/RICEWM					MN36-D	WB	4.128	
EX		2334	36/DALEWX					MN36-D	WB	3.418	
EN		2337	36/DALEWM					MN36-D	WB	3.157	
EX		2340	36/LEXGTWX					MN36-D	WB	2.451	
EN		2343	36/LEXGTWM					MN36-D	WB	2.154	
EX		2344	36/HAMLWX					MN36-D	WB	1.904	
EN		2347	36/HAMLWM					MN36-D	WB	1.837	
EX		2348	36/SNELWX1					MN36-D	WB	1.416	
EN		2349	36/SNELWM1					MN36-D	WB	1.323	
EX		2358	36/SNELWX2					MN36-D	WB	1.210	
EN		2361	36/SNELWM2					MN36-D	WB	1.055	
EX		2362	36/FAIRWX					MN36-D	WB	0.948	
EN		2365	36/FAIRWM					MN36-D	WB	0.829	
EX		2366	36/35WWX					MN36-D	WB	0.480	
EN		2369	36/CLEVWM					MN36-D	WB	0.345	
EX		2417						MN36-D	WB	5.271	
EX		2433						MN36-D	WB	5.060	
EX		2183						MN36-I	EB	0.163	
EN		2248	36/35WEM					MN36-I	EB	0.247	
EN		2249	36/CLEVEM					MN36-I	EB	0.427	
EX		2253	36/FAIREX					MN36-I	EB	0.844	
EN		2256	36/FAIREM					MN36-I	EB	0.979	
EX		2257	36/SNELEX1					MN36-I	EB	1.130	
EN		2260	36/SNELEM1					MN36-I	EB	1.218	
EX		2261	36/SNELEX2					MN36-I	EB	1.319	
EN		2262	36/SNELEM2					MN36-I	EB	1.413	
EX		2263	36/HAMLEX					MN36-I	EB	1.599	
EN		2266	36/HAMLEM					MN36-I	EB	1.650	
EX		2267	36/LEXGTEX					MN36-I	EB	2.028	
EN		2270	36/LEXGTEM					MN36-I	EB	2.425	
EX		2273	36/DALEEX					MN36-I	EB	3.075	
EN		2276	36/DALEEM					MN36-I	EB	3.445	
EX		2279	36/RICEEX					MN36-I	EB	4.112	
EN		2282	36/RICEEM					MN36-I	EB	4.399	
EN		2285						MN36-I	EB	5.055	
EN		2288						MN36-I	EB	5.310	
EX		2289	36/EDGEREX					MN36-I	EB	5.600	
EN		2292	36/EDGEREM					MN36-I	EB	5.911	

EX		2295	36/61EX1					MN36-I	EB	6.683	
EN		2298	36/61EM1					MN36-I	EB	6.733	
EX		2299	36/61EX2					MN36-I	EB	6.881	
EN		2300	36/61EM2					MN36-I	EB	7.001	
EX		2413						MN36-I	EB	5.116	
EX		2434						MN36-I	EB	4.932	
EN		1526	55/FEDBLGEM					MN55-D	SB	197.750	
EX		1542	55/FEDBLGEX					MN55-D	SB	197.669	
EX		1584	55/TH5EX					MN55-D	SB	198.217	
EX		1311	55/3STNX					MN55-I	NB	191.850	
EX		1515						MN55-I	NB	198.200	
EX		1516						MN55-I	NB	198.160	
OT		1519	55/CR205WX					MN55-I	NB	197.780	
EN		1523	55/CR205WM					MN55-I	NB	197.650	
CD		1464						MN5-D	WB	64.350	
CD		1516						MN5-D	WB	64.400	
CD		1571	5/TH55WX					MN5-D	WB	64.300	
CD		1572	5/CR205WX					MN5-D	WB	64.300	
EN		1583	5/AIRWM1					MN5-D	WB	62.910	
EX		1585	5/POSTWX					MN5-D	WB	62.590	
EN		1589	5/POSTWM					MN5-D	WB	62.354	
EX		1554	5/POSTEX					MN5-I	EB	62.360	
EN		1558	5/POSTEM					MN5-I	EB	62.585	
ML		100	IS THERE A STAT #					MN62-D	WB	112.130	
EX		126						MN62-D	WB	112.267	
EX		1073	121/62SX					MN62-D	WB		
EX		1082	62/PENWX					MN62-D	WB	110.622	
EN		1085	62/PENWM					MN62-D	WB	110.362	
EX		1086	62/XERWX					MN62-D	WB	110.122	
EN		1089	62/XERWM					MN62-D	WB	109.892	
EX		1090	62/FRAWX					MN62-D	WB	109.572	
EN		1093	62/FRAWM					MN62-D	WB	109.292	
EN		1096	62/VVRWM					MN62-D	WB	109.062	
EN		1099	62/100WM1					MN62-D	WB	108.372	
EN		1100	62/100WM2					MN62-D	WB	108.124	
EX		1105	62/TRAWX					MN62-D	WB	107.328	
EN		1106	62/TRAWM					MN62-D	WB	107.171	
EX		1110	62/GLEWX					MN62-D	WB	106.675	
EN		1111	62/GLEWM					MN62-D	WB	106.240	
EN		1117	62/169NX					MN62-D	WB	105.907	
EN		1118	62/169SX					MN62-D	WB	105.725	
EX		1126	62/SHAWX					MN62-D	WB	104.852	
EN		1129	62/SHAWM					MN62-D	WB	104.512	
EX		1158	62/34AVWX					MN62-D	WB	114.772	
EN		1161	62/34AVWM					MN62-D	WB	114.722	
EX		1162	62/28AVWX					MN62-D	WB	114.512	
EN		1165	62/28AVWM					MN62-D	WB	114.215	
EN		1166	62/TH77WM1					MN62-D	WB	113.692	
EN		1169	62/TH77WM2					MN62-D	WB	113.472	
EX		1172	62/PORTWX					MN62-D	WB	112.602	
EN		1175	62/PORTWM					MN62-D	WB	112.412	
EX		1267						MN62-D	WB	108.402	
EX		1293						MN62-D	WB	108.279	
EX		1361						MN62-D	WB	106.025	
EX		1369	62/169SM					MN62-D	WB	105.806	
EX		1454	62/TH77WX1					MN62-D	WB	113.832	
EX		1465	62/TH77WX2					MN62-D	WB	113.642	
EN		139						MN62-I	EB	112.262	

EX		166	62/LYNDEX					MN62-I	EB	111.370	
ML		167						MN62-I	EB	112.130	
EX		1022	62/SHAEX					MN62-I	EB	104.510	
EN		1023	62/SHAEM1					MN62-I	EB	104.685	
EN		1024	62/SHAEM2					MN62-I	EB	104.942	
EN		1033						MN62-I	EB	105.814	
EN		1039						MN62-I	EB	106.023	
EX		1040	62/GLEEX					MN62-I	EB	106.248	
EN		1043	62/GLEEM					MN62-I	EB	106.679	
EX		1046	62/TRAEX					MN62-I	EB	107.216	
EN		1047	62/TRAEM					MN62-I	EB	107.481	
EN		1050	62/S100EM					MN62-I	EB	108.310	
EN		1051	62/N100EM					MN62-I	EB	108.512	
EX		1056	62/VVREX					MN62-I	EB	109.102	
EX		1057	62/FRAEX					MN62-I	EB	109.322	
EN		1060	62/FRAEM					MN62-I	EB	109.602	
EX		1061	62/XEREX					MN62-I	EB	109.882	
EN		1064	62/XEREM					MN62-I	EB	110.142	
EX		1065	62/PENEX					MN62-I	EB	110.502	
EN		1068	62/PENEM					MN62-I	EB	110.632	
EX		1133	62/PORTEX					MN62-I	EB	112.422	
EN		1136	62/PORTEM					MN62-I	EB	112.602	
EX		1139	62/BLOMEX					MN62-I	EB	113.242	
EN		1140	62/BLOMEM					MN62-I	EB	113.282	
EN		1141						MN62-I	EB	113.570	
EN		1144						MN62-I	EB	113.852	
EX		1145	66/28AVEX					MN62-I	EB	114.202	
EN		1148	62/28AVEM					MN62-I	EB	114.532	
EX		1149	62/34AVEX					MN62-I	EB	114.922	
EN		1152	62/34AVEM					MN62-I	EB	114.962	
EX		1263						MN62-I	EB	108.402	
EX		1297						MN62-I	EB	108.209	
EX		1358						MN62-I	EB	105.912	
EX		1372	62/169EX					MN62-I	EB	105.706	
EX		1453						MN62-I	EB	113.682	
EX		1466	62/TH77EX					MN62-I	EB	113.462	
EN		94	35W/12THSM					MN65-D	SB	0.840	
EN		95	35W/94SM					MN65-D	SB	0.620	
EX		172	35W/W94NX					MN65-I	NB	0.600	
EX		1141						MN77-D	SB	11.328	
EN		1465						MN77-D	SB	11.380	
EN		1466						MN77-D	SB	11.218	
EX		1469	77/63STSX					MN77-D	SB	11.037	
EN		1474	77/66STSM					MN77-D	SB	10.358	
EX		1475	77/70STSX					MN77-D	SB	10.097	
EN		1478	77/70STSM					MN77-D	SB	9.977	
EX		1479	77/74STSX					MN77-D	SB	9.552	
EX		1484	77/I494SX3					MN77-D	SB	9.341	
CD		1487	77/W494SM					MN77-D	SB	9.139	
CD		1490	77/CD11					MN77-D	SB	8.850	
EN		1495	77/84STSM1					MN77-D	SB	8.460	
EN		1498	77/84STSM2					MN77-D	SB	8.290	
EN		1499	77/86STSM					MN77-D	SB	8.176	
EX		1503	77/SHAKOSX					MN77-D	SB	7.744	
EN		1507	77SHAKOSM1					MN77-D	SB	7.610	
EN		1508	77SHAKOSM2					MN77-D	SB	7.317	
CD		1529	77/E494SX					MN77-D	SB	9.039	
CD		1620						MN77-D	SB	9.300	

EX		1144						MN77-I	NB	11.224		
EX		1166	North of TH77 Milemark					MN77-I	NB	11.300		
EX		1411	77/SHAKONX					MN77-I	NB	7.634		
EN		1415	77/SHAKONM					MN77-I	NB	7.807		
EX		1416	77/86STNX					MN77-I	NB	8.166		
CD		1421	77/CD01					MN77-I	NB	8.370		
CD		1422	77/CD02					MN77-I	NB	8.350		
CD		1429						MN77-I	NB	8.800		
CD		1430	77/CD04					MN77-I	NB	8.801		
CD		1431	77/CD05					MN77-I	NB	8.802		
CD		1432	77/CD06					MN77-I	NB	8.803		
EN		1435	77/W494NM					MN77-I	NB	9.346		
EN		1438	77/74STNM					MN77-I	NB	9.586		
OT		1441	77/74STPR					MN77-I	NB	9.600		
EX		1444	77/66STNX					MN77-I	NB	10.384		
EN		1447	77/66STNM					MN77-I	NB	10.467		
EN		1448	77/63STNM					MN77-I	NB	11.025		
EN		1453	77/TH62NM					MN77-I	NB	11.360		
EN		1454						MN77-I	NB	11.301		
EN		1530						MN77-I	NB	9.035		
HX		1618						MN77-I	NB	9.139		
EX		389	100/77SX					US100-D	SB	0.539		
EN		406	100/77SM					US100-D	SB	0.364		
EN		647	100/252SM					US100-D	SB	15.900		
HR		679	100/94S1	100				US100-D	SB	15.500		
HR		680	100/94S2	100				US100-D	SB	15.500		
EX		806						US100-D	SB	0.000		
EX		807						US100-D	SB	0.001		
EN		817						US100-D	SB	0.047		
EX		818						US100-D	SB	0.116		
EX		1050						US100-D	SB	2.104		
EX		1100						US100-D	SB	2.288		
EX		1281	100/EDENSX					US100-D	SB	3.961		
EN		1285	100/EDENSM					US100-D	SB	3.482		
EX		1289	100/BENSX					US100-D	SB	2.772		
EN		1293	100/62SM1					US100-D	SB	2.198		
EN		1297	100/62SM2					US100-D	SB	2.015		
EX		1301	100/70SX					US100-D	SB	1.447		
EN		1305	100/70SM					US100-D	SB	1.000		
HR		1716						US100-D	SB	7.710		
HR		1717						US100-D	SB	7.700		
CD		1718						US100-D	SB	7.718		
EX		2060	100/DULTSX					US100-D	SB	10.017		
EN		2063	100/DULTSM					US100-D	SB	9.787		
EX		2066	100/55SX1					US100-D	SB	9.004		
EN		2067	100/55SM1					US100-D	SB	8.941		
EX		2072						US100-D	SB	8.850		
EN		2075						US100-D	SB	8.771		
EX		2076	100/GLENSX					US100-D	SB	8.472		
CD		2082	100/394SX1					US100-D	SB	7.850		
HR		2083	100/394SXH					US100-D	SB	7.851		
CD		2088						US100-D	SB	7.785		
EN		2098	100/23STSM					US100-D	SB	6.942		
EN		2102	100/27STSM					US100-D	SB	6.655		
EN		2105	100/MTKASM					US100-D	SB	6.062		
EX		2106	100/TH7SX1					US100-D	SB	6.001		
EN		2107	100/TH7SM1					US100-D	SB	5.932		
EX		2108	100/TH7SX2					US100-D	SB	5.860		

EN		2109	100/TH7SM2					US100-D	SB	5.759	
CD		2113	100/36STSX					US100-D	SB	5.510	
CD		2117	100/36STSM					US100-D	SB	5.020	
OT		2118	100/EXCLSX					US100-D	SB	5.020	
EN		2122	100/EXCLSM					US100-D	SB	4.835	
CD		2126	100/PRKSXR					US100-D	SB	7.650	
EX		2127	100/PRKSXL					US100-D	SB	7.600	
EX		70	100/77THNX					US100-I	NB	0.389	
EN		72	100/77THNM					US100-I	NB	0.558	
EN		822						US100-I	NB	0.001	
EN		823						US100-I	NB	0.123	
EX		824						US100-I	NB	0.000	
EX		825						US100-I	NB	0.047	
EX		1051						US100-I	NB	1.976	
EX		1099						US100-I	NB	2.189	
EX		1255	100/70NX					US100-I	NB	1.070	
EN		1259	100/70NM					US100-I	NB	1.400	
EN		1263	100/62NM1					US100-I	NB	2.090	
EN		1267	100/62NM2					US100-I	NB	2.316	
EN		1271	100/BENNM					US100-I	NB	2.743	
EX		1275	100/EDENNX					US100-I	NB	3.625	
EN		1276	100/EDNNM1					US100-I	NB	3.717	
EN		1277	100/EDNNM2					US100-I	NB	3.909	
EX		1749						US100-I	NB	7.800	
EX		2004	100/EXCLNX					US100-I	NB	4.862	
CD		2008	100/36STNX					US100-I	NB	5.400	
CD		2012	100/EXCLNM					US100-I	NB	5.410	
CD		2013	100/36STNM					US100-I	NB	5.420	
EX		2017	100/TH7NX1					US100-I	NB	5.784	
EN		2018	100/TH7NM1					US100-I	NB	5.860	
EX		2019	100/TH7NX2					US100-I	NB	5.975	
EN		2022	100/TH7NM2					US100-I	NB	6.020	
EX		2023	100/MTKANX					US100-I	NB	6.336	
EN		2024	100/MTKANM					US100-I	NB	6.453	
OT		2029	100/CDRCDM					US100-I	NB	7.450	
EX		2033	100/CEDRNX					US100-I	NB	7.400	
EN		2034	100/CEDRNM					US100-I	NB	7.550	
EN		2035	100/394NM1					US100-I	NB	7.718	
EN		2040	100/394NM3					US100-I	NB	7.950	
EX		2041	100/GLENNX					US100-I	NB	8.186	
EX		2047	100/55NX1					US100-I	NB	8.771	
EN		2048	100/55NM1					US100-I	NB	8.859	
EX		2049	100/55NX2					US100-I	NB	8.945	
EN		2052	100/55NM2					US100-I	NB	9.020	
EN		2057	100/DULTNM					US100-I	NB	10.018	
EX		2058	100/DULTNX					US100-I	NB	9.787	
CD		2084	100/394SX2					US100-I	NB	7.816	
CD		1814	12/494WM2					US12-D	WB	156.850	
CD		1815	12/494WM1					US12-D	WB	156.821	
CD		1816	12/CARLWX					US12-D	WB	156.520	
CD		1819	12/CARLWM1					US12-D	WB	156.410	
EN		1823	12/CARLWM2					US12-D	WB	156.228	
EX		1824	12/GLEAWX1					US12-D	WB	155.626	
EX		1828	12/GLEAWX2					US12-D	WB	155.414	
EX		1829	12/CENTWX					US12-D	WB	155.049	
EN		1833	12/CENTWM					US12-D	WB	154.724	
CD		1868						US12-D	WB	156.956	
HO		1629	12/GLEAEMH					US12-I	EB	155.590	

EX		1630	12/CENTEX					US12-I	EB	154.765	
EN		1634	12/CENTEM					US12-I	EB	155.071	
EX		1639	12/CARLEX1					US12-I	EB	156.225	
CD		1640	12/CARLEX2					US12-I	EB	156.397	
CD		1644	12/CARLEM					US12-I	EB	156.521	
CD		1648	12/494EM					US12-I	EB	156.954	
CD		1871						US12-I	EB	156.814	
EX		942						US169-D	SB	137.800	
EX		943						US169-D	SB	138.080	
EN		944						US169-D	SB	137.920	
EN		945						US169-D	SB	137.640	
EN		1033	169/62EM1					US169-D	SB	124.320	
EX		1118						US169-D	SB	124.600	
EN		1368	169/BRENSM					US169-D	SB	124.800	
EN		1369						US169-D	SB	124.415	
EN		1372						US169-D	SB	124.167	
EX		1375	169/VVRSX					US169-D	SB	123.325	
EN		1378	169/VVRSM					US169-D	SB	122.985	
EX		1680	169/394SX					US169-D	SB	130.210	
EX		1786						US169-D	SB	130.463	
EX		1954	169/55SX1					US169-D	SB	131.140	
EN		1957	169/55SM1					US169-D	SB	131.000	
EX		1958	169/55SX2					US169-D	SB	130.900	
EN		1961	169/55SM2					US169-D	SB	130.810	
EX		1962	169/BETYSX					US169-D	SB	130.640	
EN		1963	169/BETYSM					US169-D	SB	130.565	
EN		1966	169/394SM1					US169-D	SB	130.320	
EN		1969	169/394SM2					US169-D	SB	130.080	
EX		1972	169/CEDRSX					US169-D	SB	128.797	
EN		1975	169/CEDRSM					US169-D	SB	128.756	
EX		1976	169/MTKASX					US169-D	SB	128.399	
EN		1979	169/MTKASM					US169-D	SB	128.202	
EX		1983	169/36STSX					US169-D	SB	128.030	
EX		1984	169/TH7SX					US169-D	SB	127.760	
EN		1987	169/TH7SM1					US169-D	SB	127.506	
EN		1990	169/TH7SM2					US169-D	SB	127.270	
EX		1991	169/EXCLSX					US169-D	SB	126.957	
EN		1994	169/EXCLSM					US169-D	SB	126.656	
EX		1995	169/7STSX					US169-D	SB	126.080	
EN		1998	169/7STSM					US169-D	SB	125.820	
EN		2960						US169-D	SB	138.710	
EN		2961						US169-D	SB	136.180	
EN		2962						US169-D	SB	134.481	
EX		2965						US169-D	SB	131.760	
EN		2968						US169-D	SB	131.420	
EX		2971						US169-D	SB	132.730	
EN		2974						US169-D	SB	132.670	
EX		2977						US169-D	SB	133.810	
HN		2980						US169-D	SB	133.520	
EX		2982						US169-D	SB	134.641	
EX		2985						US169-D	SB	134.360	
EN		2986						US169-D	SB	134.190	
EX		2987						US169-D	SB	135.310	
EN		2990						US169-D	SB	135.001	
EX		2993						US169-D	SB	136.340	
EX		2996						US169-D	SB	136.071	
EN		2997						US169-D	SB	135.910	
EX		3000						US169-D	SB	137.000	

EN		3003					US169-D	SB	136.950	
EN		3006					US169-D	SB	137.920	
EX		3009					US169-D	SB	138.880	
EN		3014					US169-D	SB	138.440	
EN		946					US169-I	NB	137.780	
EN		947					US169-I	NB	138.070	
EX		948					US169-I	NB	137.930	
EX		949					US169-I	NB	137.630	
EN		1039	169/62EM2				US169-I	NB	124.160	
EX		1117					US169-I	NB	124.415	
EX		1352	169/VVRNX				US169-I	NB	122.950	
EN		1355	169/VVRNM				US169-I	NB	123.250	
EN		1358	169/62NM1				US169-I	NB	124.340	
EN		1361	169/62NM2				US169-I	NB	124.560	
EN		1364	169/BRENNM				US169-I	NB	125.192	
EX		1681	169/394NX				US169-I	NB	130.074	
EX		1783					US169-I	NB	130.331	
EX		1901	169/7STNX				US169-I	NB	125.830	
EN		1904	169/7STNM				US169-I	NB	126.090	
EX		1905	169/EXCLNX				US169-I	NB	126.704	
EN		1908	169/EXCLNM				US169-I	NB	126.957	
EX		1909	169/TH7NX				US169-I	NB	127.265	
EN		1914	169/TH7NM				US169-I	NB	127.672	
EN		1917	169/36STNM				US169-I	NB	127.974	
HN		1918	169/394NMH				US169-I	NB	130.446	
EX		1921	169/MTKANX				US169-I	NB	128.199	
EN		1924	169/MTKANM				US169-I	NB	128.448	
EX		1925	169/CEDRNX				US169-I	NB	128.755	
EN		1928	169/CEDRNM				US169-I	NB	128.798	
EX		1929	169/22STNX				US169-I	NB	129.125	
EN		1932	169/22STNM				US169-I	NB	129.392	
EN		1935	169/394NM1				US169-I	NB	130.220	
EX		1939	169/BETYNX				US169-I	NB	130.565	
EN		1940	169/BETYNM				US169-I	NB	130.640	
EX		1941	169/55NX1				US169-I	NB	130.780	
EN		1944	169/55NM1				US169-I	NB	130.900	
EX		1945	169/55NX2				US169-I	NB	131.010	
EN		1948	169/55NM2				US169-I	NB	131.110	
EN		3016					US169-I	NB	136.070	
EN		3017					US169-I	NB	134.360	
EX		3020					US169-I	NB	138.500	
EN		3025					US169-I	NB	138.860	
EX		3030					US169-I	NB	136.820	
EN		3033					US169-I	NB	136.870	
EX		3036					US169-I	NB	135.900	
EX		3039					US169-I	NB	136.170	
EN		3040					US169-I	NB	136.330	
EX		3043					US169-I	NB	135.000	
EN		3046					US169-I	NB	135.300	
EX		3047					US169-I	NB	134.200	
EX		3050					US169-I	NB	134.480	
EN		3051					US169-I	NB	134.640	
EX		3052					US169-I	NB	133.490	
EN		3055					US169-I	NB	133.770	
EX		3058					US169-I	NB	132.430	
EN		3061					US169-I	NB	132.770	
EX		3064					US169-I	NB	131.380	
EN		3067					US169-I	NB	131.730	

EN		1121	212/TH62WM					US212-D	SB	161.439	
EX		1336	212/SHADWX					US212-D	SB	161.184	
EN		1339	212/SHADSM					US212-D	SB	160.806	
EX		1342	212/BRYLKWX					US212-D	SB	160.242	
EN		1343	212/BRYLKWM					US212-D	SB	160.060	
EX		1031	212/TH62EX					US212-I	NB	161.415	
EX		1324	212/VVREX					US212-I	NB	159.941	
EN		1325	212/VVREM					US212-I	NB	160.104	
EX		1328	212/SHADEX					US212-I	NB	160.864	
EN		1331	212/SHADEM					US212-I	NB	161.196	

Appendix D - RDD Server Source Code (30-second)

D.1 CORBA Files

D.1.1 RDD.idl

```
////////////////////////////////////////////////////////////////
// RDD.idl
//
// IDL file for ITS Real-time Data Distributor (RDD)
//
// Written by Ken Reynhout, 12/6/97
//
////////////////////////////////////////////////////////////////

// Note: In CORBA short = 16 bits
//                long  = 32 bits

// Typedefs
// 5-minute detector data
struct Detector {
        unsigned short id;          // 1 to 3500
        unsigned short time;        // formatted hhmm, 0000 to 2359
        unsigned short volume;      // 0 to 255
        float occupancy;            // 0 to 100
        short status;               // communication status, -4 to 3
        short validity;             // error flag, -4 to 3
};
typedef Detector AllDetectors[3500];

// 30-second station data
struct Station {
        unsigned short id;          // 1 to 864
        unsigned short volume;      // 0 to 63
        float occupancy;            // 0 to 100
        short status;               // communication status, -4 to 3
};
typedef Station AllStations[864];

// 30-second ramp meter data
struct RampMeter {
        unsigned short id;          // 0 to 1000
        unsigned short fieldRate;   // as reported by controller, 0 to 7
        unsigned short computedRate;    // from TMC ramp metering algorithm, 0-7
        unsigned short overrideRate;    // from operator, 0 to 7
        short status;                   // communication status, -4 to 3
        unsigned short mode;            // 0 to 12 (see below)
};
typedef RampMeter AllRampMeters[1000]; //???

interface RDD {

        // operations
        oneway void deliverDetector (in Detector package);
        oneway void deliverStation (in Station package);
        oneway void deliverRampMeter (in RampMeter package);
```

```
        oneway void deliverAllDetectors (in AllDetectors package);
        oneway void deliverAllStations (in AllStations package);
        oneway void deliverAllRampMeters (in AllRampMeters package);

};
// Ramp meter modes
// 0      Inactive, not metering
// 1      Operator has entered override
// 2      Minimum rate
// 3      Volume computed rate
// 4      Occupancy computed rate
// 5      Meter shutdown mode
// 6      Volume station failure
// 7      Communication failure
// 8      Startup (system in transition)
// 9      Time of day metering
// 10     Local manual rate at 170 controller
// 11     Speedlight mode
// 12     SCATS computer controlling ramp
```

D.1.2 RDD.hh

Created by the Orbix CORBA compiler, not included here.

D.1.3 RDDC.cpp

Created by the Orbix CORBA compiler, not included here.

D.1.4 RDDS.cpp

Created by the Orbix CORBA compiler, not included here.

D.2 30-second Loop Server Files

D.2.1 Const.h

```
// maximum number of detectors
#define NUM_DETECTORS     3500
#define NUM_STATIONS      864

/*--- MNEMONICS FOR RESPONSES FROM THE PIPE SERVER ----*/
#define RESPONSE_SUCCESS  0     /* SUCCESSFUL OPERATION */
#define RESPONSE_FAILED   1     /* FAILED OPERATION */
#define RESPONSE_STARTED  100   /* SERVER SIDE STARTED */

/*--- COMMANDS WHICH ARE SENT TO THE PIPE SERVER ---*/
#define CMD_WRITE       2     /* WRITE COMMAND */
#define CMD_READ        3     /* READ COMMAND */
#define CMD_SYSTEM      4     /* SYSTEM COMMAND */
#define CMD_SHUTDOWN    5     /* SHUTDOWN COMMAND */
#define CMD_BYEBYE      6
#define CMD_NEWCON      7
#define CMD_FAILURE     8
#define CMD_PAUSE       9
#define CMD_UNAVAILABLE 10
```

```
#define _MIN_PIPE_BUFFER   1024   /* USED FOR PIPEDATA TYPEDEF */


#define TRUE   1
#define FALSE  0

/*---------------------*/
/* Constant definitions */
/*---------------------*/

#define NPIPE_30S        1
#define NPIPE_5M         2
#define NPIPE_RM         3
#define IPLISTENER       4

#define SCOUNT 27
#define DCOUNT 119
```

D.2.2 CIntsock.h

```
//
// This is a part of the Microsoft Foundation Classes C++ library.
// Copyright (C) 1992-1995 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.

#ifndef __CHATSOCK_H__
#define __CHATSOCK_H__

class CLoopServDoc;

class CChatSocket : public CSocket
{
        DECLARE_DYNAMIC(CChatSocket);

// Construction
public:
        CChatSocket(CLoopServDoc* pDoc);

// Operations
public:
        CLoopServDoc* m_pDoc;

// Implementation
protected:
        virtual void OnReceive(int nErrorCode);
};

#endif // __CHATSOCK_H__
```

D.2.3 Datatype.h

```
#ifndef __DATATYPE
#define __DATATYPE

/*------------------*/
/* Type definitions */
/*------------------*/
 typedef struct
   {
   CString IPAddress1;
          CString IPAddress2;
          int  autostart;
          int  restart;
          CString userName;
   CString userLocation;
   CString userPhone;
          CString userEmail;
          CString userApp;
  } PROGRAM_INFO;

 typedef struct
   {
   short usCmd;
   short dataReq;
   long m_lConnection;
   long m_lStatCount;
   long m_lPacketSize;
   char userName[25];
   char userLocation[25];
   char userPhone[20];
          char userEmail[40];
          char userApp[20];
  } COMMAND;

/* ---------------------------
/* Client login pipe messages
*/
typedef struct {
  CHAR      cMsgType;
  CHAR      szClientID[16];
  CHAR      szPassWord[16];
} LoginMsg_t, *pLoginMsg_t;


/* ---------------------------
// Response to Login request
*/
typedef struct {
  CHAR    cMsgType;
  ULONG   ulSecurity;
  CHAR    szPipeName[9];
} LoginResponse_t, *pLoginResponse_t;


/* ---------------------------
// Client data request packet
```

```
*/
typedef struct {
  CHAR    cMsgType;
  ULONG   ulReq;
} CliDataReqMsg_t, *pCliDataReqMsg_t;


/* ------------------------------------------
  Vol and Occ for a single interval / station
*/
typedef struct {
  USHORT  Vol: 6;      // 30-Second volume (0 to 63)
  USHORT  Occ: 7;      // 30-Second Occupancy (0 to 100)
  SHORT   Status: 3;   // Communication status
} Int30SecData_t, *pInt30SecData_t;

/* ------------------------------------------
// 5-Minute detector data for a single detector
*/
typedef struct {
  UCHAR   Vol;         // 5-Minute Volume (0 to 255)
  SHORT   Occ: 8;      // 5-Minute Occupancy (0 to 100)
  SHORT   Status: 4;   // Communication status (-8 to +7)
  SHORT   Flag: 4;     // Detector flag code (-8 to +7)
} Int5MinData_t, *pInt5MinData_t;


// ------------------------------------------
// Structure for Ramp Meter status information
//
typedef struct {
  USHORT    Field: 3;     // Field rate as reported by controller
  USHORT    Computed: 3;  // Rate computed by ramp metering algorithm
  USHORT    Override: 3;  // Override rate from operator
  SHORT     Status: 3;    // Communication status
  USHORT    Mode: 4;      // Ramp metering mode
} AllRates_t, *pAllRates_t;


// ------------------------------------------------
// Data type Count message passed to client programs
//
typedef struct {
  CHAR    cMsgType;
  ULONG   ulCount;
} CountMsg_t, *pCountMsg_t;


// ------------------------------------------
// Ramp information passed to client programs
//
typedef struct {
  CHAR    cMsgType;
  CHAR    szRampID[4];    // 3-character ramp id
  USHORT  usLine;         // line number for this ramp
  USHORT  usDrop;         // drop address for this ramp
```

```
  USHORT  usRamp;         // ramp 1 or ramp 2
  CHAR    szFrom[21];     // street ramp
  CHAR    szFreeway[21];  // freeway name
  USHORT  usRate[2];      // default rate AM/PM
} MeterInfo_t, *pMeterInfo_t;

typedef struct {
  USHORT  usSecond: 6;  // Second (0-59)
  USHORT  usMinute: 6;  // Minute (0-59)
  USHORT  usMonth: 4;   // Month (1-12)
  USHORT  usHour: 5;    // Hour (0-23)
  USHORT  usDay: 5;     // Day (1-31)
  USHORT  usYear: 6;    // Year (0-63 + 1980)
} DateTime_t, *pDateTime_t;

#endif
```

D.2.4 stdafx.h (for precompiling)

```
// include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows headers

#include <afxwin.h>       // MFC core and standard components
#include <afxext.h>       // MFC extensions
#include <afxdb.h>                      // MFC ODBC database classes

#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC support for Windows 95 Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT

#include <afxsock.h>            // MFC socket extensions

#include <iostream.h>

#include <orbixTalk.h>          // OrbixTalk definitions
```

D.2.5 resource.h (for user interface resources)

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by LoopServ.rc
//
#define IDD_ABOUTBOX                100
#define IDD_LOOPSERV_FORM           101
#define IDP_FAILED_OPEN_DATABASE    103
#define IDP_SOCKETS_INIT_FAILED     104
#define IDR_MAINFRAME               128
#define IDR_LOOPSERV_TYPE           129
#define ID_FILE_CONNECT             32771
#define IDM_FILE_CONNECT            32771
#define IDM_REGISTER_TALKERS        32772

// Next default values for new objects
```

```
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS                1
#define _APS_NEXT_RESOURCE_VALUE        130
#define _APS_NEXT_COMMAND_VALUE         32773
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

D.2.6 Mainfrm.h

```
// interface of the CLoopServFrame class
//
/////////////////////////////////////////////////////////////////////////////
#ifndef __LoopServFRM
#define __LoopServFRM

class CLoopServFrame : public CFrameWnd
{
protected: // create from serialization only
        CLoopServFrame();
        DECLARE_DYNCREATE(CLoopServFrame)

// Attributes
public:
        BOOL m_bREGISTERED;
        BOOL m_bCONNECTED;

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CLoopServFrame)
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CLoopServFrame();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
        CStatusBar  m_wndStatusBar;

// Generated message map functions
protected:
        //{{AFX_MSG(CLoopServFrame)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
        afx_msg void OnFileConnect();
        afx_msg void OnUpdateFileConnect(CCmdUI* pCmdUI);
        afx_msg void OnRegisterTalkers();
```

```
        afx_msg void OnUpdateRegisterTalkers(CCmdUI* pCmdUI);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

#endif
```

D.2.7 LoopServ.h

```
// LoopServ.h : main header file for the LoopServ application
//

#ifndef __AFXWIN_H__
        #error include 'stdafx.h' before including this file for PCH
#endif

#include "Stdafx.h"
#include "rdd.hh"
#include "CONST.h"
#include "resource.h"       // main symbols

static FILE *test_fp; // for debugging

extern CStatusBar* e_status;
extern OrbixTalk_ptr otalk;                                          // OrbixTalk object
extern RDD_var LoopObject[NUM_STATIONS];  // OrbixTalk topics to talk on
extern RDD_var LoopObjectAll;  // OrbixTalk topics to talk on

/////////////////////////////////////////////////////////////////////////////
// CLoopServApp:
// See LoopServ.cpp for the implementation of this class
//

class CLoopServApp : public CWinApp
{
public:
        CLoopServApp();

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CLoopServApp)
        public:
        virtual BOOL InitInstance();
        //}}AFX_VIRTUAL

// Implementation

        //{{AFX_MSG(CLoopServApp)
        afx_msg void OnAppAbout();
                // NOTE - the ClassWizard will add and remove member functions here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
```

D.2.8 LoopServDoc.h

```
// LoopServDoc.h : interface of the CLoopServDoc class
//
/////////////////////////////////////////////////////////////////////////////
#ifndef __LoopServDOC
#define __LoopServDOC

#include "clntsock.h"
#include "const.h"
#include "datatype.h"

class CLoopServDoc : public CDocument
{
protected: // create from serialization only
        CLoopServDoc();
        DECLARE_DYNCREATE(CLoopServDoc)

// Attributes
public:

   COMMAND*          m_pCommandPacket;
        Int30SecData_t* DataPacket;
        char*                Data;
        DateTime_t*          TimeStamp;
        CChatSocket*    m_pSocket;
        CSocketFile*    m_pFile;
        CFile*               pOutput;
        CTime                theTime;
        BOOL                 m_bCONNECTED;

private:

   BOOL m_bFIRST_MSG;
        int  m_iPacketCounter;
        long m_lConnection;
   long m_lStatCount;
   long m_lPacketSize;
        //int  m_iDay, m_iMonth, m_iYear;
        //int  m_iHour, m_iMinute, m_iSecond;
        //int  m_iTime,m_iDdate;
        int  m_iDataRead;
        int  DataTotal;
        int  LastTime;

// Operations
public:

        BOOL ConnectSocket();
        void ProcessPendingRead();
        void SendMsg();
        void ReceiveMsg();
   BOOL RegisterAllTalkers();
   RDD_ptr myRegisterTalker (const char* sLoop);

// Overrides
        // ClassWizard generated virtual function overrides
```

```
        //{{AFX_VIRTUAL(CLoopServDoc)
        public:
        virtual BOOL OnNewDocument();
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CLoopServDoc();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
        //{{AFX_MSG(CLoopServDoc)
                // NOTE - the ClassWizard will add and remove member functions here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////

#endif
```

D.2.9 LoopServFrame.h

```
// LoopServFrame.h : interface of the CLoopServFrame class
//
/////////////////////////////////////////////////////////////////////////////
#ifndef __LoopServFRM
#define __LoopServFRM

class CLoopServFrame : public CFrameWnd
{
protected: // create from serialization only
        CLoopServFrame();
        DECLARE_DYNCREATE(CLoopServFrame)

// Attributes
public:
        BOOL m_bREGISTERED;
        BOOL m_bCONNECTED;

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CLoopServFrame)
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        //}}AFX_VIRTUAL

// Implementation
public:
```

```
        virtual ~CLoopServFrame();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
        CStatusBar  m_wndStatusBar;

// Generated message map functions
protected:
        //{{AFX_MSG(CLoopServFrame)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
        afx_msg void OnFileConnect();
        afx_msg void OnUpdateFileConnect(CCmdUI* pCmdUI);
        afx_msg void OnRegisterTalkers();
        afx_msg void OnUpdateRegisterTalkers(CCmdUI* pCmdUI);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////

#endif
```

D.2.10 LoopServView.h

```
// LoopServView.h : interface of the CLoopServView class
//
/////////////////////////////////////////////////////////////////////////////
#ifndef __LoopServVIEW
#define __LoopServVIEW

class CLoopServView : public CView
{
protected: // create from serialization only
        CLoopServView();
        DECLARE_DYNCREATE(CLoopServView)

public:
        //{{AFX_DATA(CLoopServView)
        enum{ IDD = IDD_LOOPSERV_FORM };
                // NOTE: the ClassWizard will add data members here
        //}}AFX_DATA

// Attributes
public:


// Operations
public:

        CLoopServDoc* GetDocument();

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CLoopServView)
        public:
```

```
        virtual void OnDraw(CDC* pDC);  // overridden to draw this view
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        virtual void OnInitialUpdate(); // called first time after construct
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CLoopServView();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
        //{{AFX_MSG(CLoopServView)
                // NOTE - the ClassWizard will add and remove member functions here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in LoopServView.cpp
inline CLoopServDoc* CLoopServView::GetDocument()
   { return (CLoopServDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////

#endif
```

D.2.11 Clntsock.cpp

```
// chatsock.cpp : implementation file
//
// This is a part of the Microsoft Foundation Classes C++ library.
// Copyright (C) 1992-1995 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.

#include "stdafx.h"
#include "LoopServ.h"
#include "clntsock.h"
#include "LoopServDoc.h"

#include <stddef.h>

#ifdef _DEBUG
#undef THIS_FILE
```

```
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

IMPLEMENT_DYNAMIC(CChatSocket, CSocket)

CChatSocket::CChatSocket(CLoopServDoc* pDoc)
{
        m_pDoc = pDoc;
}

void CChatSocket::OnReceive(int nErrorCode)
{
        CSocket::OnReceive(nErrorCode);

        m_pDoc->ProcessPendingRead();
}
```

D.2.12 Stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
//       LoopServ.pch will be the pre-compiled header
//       stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

D.2.13 Mainfrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#ifndef __LoopServFRM
#define __LoopServFRM

#include "stdafx.h"
#include "LoopServ.h"
#include "LoopServDoc.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
        //{{AFX_MSG_MAP(CMainFrame)
        ON_WM_CREATE()
        ON_COMMAND(IDM_FILE_CONNECT, OnFileConnect)
        ON_UPDATE_COMMAND_UI(IDM_FILE_CONNECT, OnUpdateFileConnect)
        ON_COMMAND(IDM_REGISTER_TALKERS, OnRegisterTalkers)
        ON_UPDATE_COMMAND_UI(IDM_REGISTER_TALKERS, OnUpdateRegisterTalkers)
        //}}AFX_MSG_MAP
```

```
END_MESSAGE_MAP()

static UINT indicators[] =
{
        ID_SEPARATOR,           // status line indicator
        ID_SEPARATOR,           // status line indicator
};

//////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
        :m_bREGISTERED(FALSE), m_bCONNECTED(FALSE)
{
        // TODO: add member initialization code here

}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
                return -1;

        if (!m_wndStatusBar.Create(this) ||
                !m_wndStatusBar.SetIndicators(indicators,
                  sizeof(indicators)/sizeof(UINT)))
        {
                TRACE0("Failed to create status bar\n");
                return -1;      // fail to create
        }

        m_wndStatusBar.SetPaneInfo(0,0,SBPS_STRETCH,0);  // Make 1st Pane Beveled
  m_wndStatusBar.SetPaneInfo(1,NULL,NULL,105);

  char StatText[20];
  sprintf(StatText, "Not Connected");

  e_status = &m_wndStatusBar;
  e_status->SetPaneText(1, StatText);

        return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs

        return CFrameWnd::PreCreateWindow(cs);
}

//////////////////////////////////////////////////////////////////////////
```

```
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
        CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
        CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers

//////////////////
// OnFileConnect() called when Connect option is chosen under File menu
void CMainFrame::OnFileConnect()
{
        CLoopServDoc* pDoc;
        char StatText[20];

        pDoc = (CLoopServDoc*) GetActiveDocument();

        if (pDoc->ConnectSocket()) {
                // connection to TMC data server was made successfully
//              pDoc->m_pCommandPacket->dataReq = NPIPE_5M;
                pDoc->m_pCommandPacket->dataReq = NPIPE_30S;
                pDoc->m_pCommandPacket->usCmd = CMD_NEWCON;
//              sprintf(pDoc->CommandPacket-
>userName,"%s",ProgramInfo.userName.GetBuffer(25));
//              sprintf(pDoc->CommandPacket-
>userLocation,"%s",ProgramInfo.userLocation.GetBuffer(25));
//              sprintf(pDoc->CommandPacket-
>userPhone,"%s",ProgramInfo.userPhone.GetBuffer(20));
//              sprintf(pDoc->CommandPacket-
>userEmail,"%s",ProgramInfo.userEmail.GetBuffer(40));
//              sprintf(pDoc->CommandPacket->userApp,"Spim");
                sprintf(pDoc->m_pCommandPacket->userName,"Greg");
                sprintf(pDoc->m_pCommandPacket->userLocation,"WE");
                sprintf(pDoc->m_pCommandPacket->userPhone,"22");
                sprintf(pDoc->m_pCommandPacket->userEmail,"22");
                sprintf(pDoc->m_pCommandPacket->userApp,"Jim");
                pDoc->SendMsg();

                sprintf(StatText, "Connected to Server");
          e_status->SetPaneText(1, StatText);

                m_bCONNECTED = TRUE;
                pDoc->m_bCONNECTED = TRUE;
        }
        else {
                // connection to TMC data server was not made successfully
```

```
                sprintf(StatText, "Server Not Found");

                e_status = &m_wndStatusBar;
                e_status->SetPaneText(1, StatText);
        }

}

void CMainFrame::OnUpdateFileConnect(CCmdUI* pCmdUI)
{
        pCmdUI->Enable(!m_bCONNECTED);
}

// OnFileConnect() called when Connect option is chosen under File menu
void CMainFrame::OnRegisterTalkers()
{
        CLoopServDoc* pDoc;
        char StatText[20];

        pDoc = (CLoopServDoc*) GetActiveDocument();

        // register the talker objects
        pDoc->RegisterAllTalkers();  // This is driving me nuts!!!!!

        sprintf(StatText, "Talkers Registered");
  e_status->SetPaneText(1, StatText);

        m_bREGISTERED = TRUE;

}

void CMainFrame::OnUpdateRegisterTalkers(CCmdUI* pCmdUI)
{
        pCmdUI->Enable(!m_bREGISTERED);
}

#endif
```

D.2.14 LoopServ.cpp

```
// LoopServ.cpp : Defines the class behaviors for the application.
//
/////////////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "LoopServ.h"
#include "LoopServFrame.h"
#include "LoopServDoc.h"
#include "LoopServView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

CStatusBar* e_status;
```

```
OrbixTalk_ptr otalk = NULL;                                    // OrbixTalk object
RDD_var LoopObject[NUM_STATIONS];  // OrbixTalk topics to talk on
RDD_var LoopObjectAll;  // OrbixTalk topics to talk on

/////////////////////////////////////////////////////////////////////////////
// CLoopServApp

BEGIN_MESSAGE_MAP(CLoopServApp, CWinApp)
        //{{AFX_MSG_MAP(CLoopServApp)
        ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CLoopServApp construction

CLoopServApp::CLoopServApp()
{
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CLoopServApp object

CLoopServApp theApp;

UINT StartOrbix(LPVOID pParam)
{
        // Process Orbix events
        CORBA(Orbix).processEvents(CORBA(Orbix).INFINITE_TIMEOUT);
        return 0;
}

/////////////////////////////////////////////////////////////////////////////
// CLoopServApp initialization

BOOL CLoopServApp::InitInstance()
{
        // Standard initialization
        // If you are not using these features and wish to reduce the size
        //  of your final executable, you should remove from the following
        //  the specific initialization routines you do not need.

#ifdef _AFXDLL
        Enable3dControls();                        // Call this when using MFC in a shared DLL
#else
        Enable3dControlsStatic(); // Call this when linking to MFC statically
#endif

        LoadStdProfileSettings(0);  // Load standard INI file options (including MRU)

        // Register the application's document templates.  Document templates
        //  serve as the connection between documents, frame windows and views.
```

```
        CSingleDocTemplate* pDocTemplate;
        pDocTemplate = new CSingleDocTemplate(
                IDR_MAINFRAME,
                RUNTIME_CLASS(CLoopServDoc),
                RUNTIME_CLASS(CLoopServFrame),      // main SDI frame window
                RUNTIME_CLASS(CLoopServView));
        AddDocTemplate(pDocTemplate);

        // Parse command line for standard shell commands, DDE, file open
        CCommandLineInfo cmdInfo;
        ParseCommandLine(cmdInfo);

        // Dispatch commands specified on the command line
        if (!ProcessShellCommand(cmdInfo))
                return FALSE;

        // Initialize OrbixTalk
  try {
                otalk = OrbixTalk::initialise ();
        } catch (CORBA(SystemException) &sysEx) {
                cerr << "Unexpected system exception, exitting" << endl;
                cerr << &sysEx;
                exit(1);
        } catch(...) {
                cerr << "Unexpected exception " << endl;
                exit(1);
        }

        AfxBeginThread(StartOrbix, NULL);

        if (!AfxSocketInit())
        {
                AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
                return FALSE;
        }

        return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
```

```
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
                // No message handlers
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
                // No message handlers
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CLoopServApp::OnAppAbout()
{
        CAboutDlg aboutDlg;
        aboutDlg.DoModal();
}


/////////////////////////////////////////////////////////////////////////////
// CLoopServApp commands
```

D.2.15 LoopServDoc.cpp

```
// LoopServDoc.cpp : implementation of the CLoopServDoc class
//

#include "stdafx.h"
#include "LoopServ.h"
#include "LoopServDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
```

```
// CLoopServDoc

IMPLEMENT_DYNCREATE(CLoopServDoc, CDocument)

BEGIN_MESSAGE_MAP(CLoopServDoc, CDocument)
        //{{AFX_MSG_MAP(CLoopServDoc)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CLoopServDoc construction/destruction

CLoopServDoc::CLoopServDoc()
{
        m_pSocket = NULL;
        m_pFile = NULL;
        m_bCONNECTED = FALSE;
        m_bFIRST_MSG = TRUE;
        m_pCommandPacket = (COMMAND *)new(COMMAND);
        m_iPacketCounter = 0;
        DataTotal = 0;
        m_iDataRead = 0;
        LastTime = 300;

}

CLoopServDoc::~CLoopServDoc()
{
        if (m_bCONNECTED) {
                delete m_pSocket;
                delete m_pFile;
        }
        if (!m_bFIRST_MSG) {
                free(Data);
        }
        delete m_pCommandPacket;
}

BOOL CLoopServDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;

        // TODO: add reinitialization code here
        // (SDI documents will reuse this document)

        SetTitle("30 Second");

        return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CLoopServDoc diagnostics
```

```
#ifdef _DEBUG
void CLoopServDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CLoopServDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CLoopServDoc commands

BOOL CLoopServDoc::ConnectSocket()
{

	m_pSocket = new CChatSocket(this);

	if (!m_pSocket->Create())
	{
		delete m_pSocket;
		m_pSocket = NULL;
		return FALSE;
	}

//	while (!m_pSocket->Connect("151.111.254.78", 1500)) // At WE
	while (!m_pSocket->Connect("151.111.254.238", 1500)) // At TMC (alternate)
	{
		delete m_pSocket;
		m_pSocket = NULL;
		return FALSE;
	}

	m_pFile = new CSocketFile(m_pSocket);

	m_bCONNECTED = TRUE;

	return TRUE;
}

void CLoopServDoc::ProcessPendingRead()
{
	ReceiveMsg();  // To allow several reads, add later
}

char* month_str[] = {"JAN", "FEB", "MAR", "APR", "MAY", "JUN",
		"JUL", "AUG", "SEP", "OCT", "NOV", "DEC"};

void CLoopServDoc::SendMsg()
{
	m_pFile->Write((COMMAND *) m_pCommandPacket, sizeof(COMMAND));
}

void CLoopServDoc::ReceiveMsg()
```

```
{
        char StatText[30];
        int  LapCounter = 0;
        //char NameBuff[20];
        //char NameBuff1[20];
        //char time1[10];

        //int speed;

        if (m_bFIRST_MSG)  {

                m_pFile->Read((COMMAND *) m_pCommandPacket, sizeof(COMMAND));
                m_lConnection = m_pCommandPacket->m_lConnection;
                m_lStatCount = m_pCommandPacket->m_lStatCount;
                m_lPacketSize = m_pCommandPacket->m_lPacketSize;

                //int tSize = sizeof(DateTime_t);

                Data = (char *) malloc(m_lPacketSize);
                DataPacket = ((Int30SecData_t *)(Data));
                //DataPacket = ((Int5MinData_t *)(Data+tSize));
                //TimeStamp = ((DateTime_t *) Data);

                m_bFIRST_MSG = FALSE;

                sprintf(StatText, "Waiting to receive...");
                e_status->SetPaneText(1, StatText);

        }
        else
        {
                sprintf(StatText, "Collecting data...");
                e_status->SetPaneText(1, StatText);

                m_iDataRead = m_pFile->Read((char *) (Data+DataTotal), (m_lPacketSize-DataTotal));
                DataTotal = DataTotal + m_iDataRead;

                if (DataTotal == (m_lPacketSize-1))  {
                        // Initialize Time Stamp
                        /*
                        m_iHour = TimeStamp->usHour;
                        m_iMinute = TimeStamp->usMinute;
                        m_iTime = (TimeStamp->usHour*12)+(TimeStamp->usMinute/5);
                        sprintf (time1, "%02d%02d", TimeStamp->usHour, TimeStamp->usMinute);

                        m_iYear = TimeStamp->usYear;
                        m_iYear = m_iYear + 80;
                        m_iMonth = TimeStamp->usMonth;
                        m_iDay = TimeStamp->usDay;
                        sprintf(NameBuff,"%2d%2d%2d",m_iDay,m_iMonth,m_iYear);

                        m_iDdate = atoi(NameBuff);
                        sprintf(NameBuff1,"%2d-%3s-%2d",m_iDay,month_str[m_iMonth],m_iYear);
                        */
                        DataTotal = 0;
                        m_iDataRead = 0;
```

```
                        // Structure to hold the data to be sent
                        AllStations LoopData;

                        while (LapCounter < m_lStatCount)  {

                          //  nothing done with speed;
                                //if (DataPacket[LapCounter].Occ != 0)  {
                                //        speed =
(5*(DataPacket[LapCounter].Vol))/(DataPacket[LapCounter].Occ);
                                //}
                                //else  {
                                //        speed = 0;
                                //}

                                LoopData[LapCounter].id = LapCounter+1;
                                // for 5 minute data
                                //LoopData[LapCounter].time = m_iTime;
                                LoopData[LapCounter].volume = DataPacket[LapCounter].Vol;
                                LoopData[LapCounter].occupancy = DataPacket[LapCounter].Occ;
                                LoopData[LapCounter].status = DataPacket[LapCounter].Status;
                                // for 5 minute data
                                //LoopData[LapCounter].validity = DataPacket[LapCounter].Flag;

                                ++LapCounter;
                        } // end while (LapCounter < dCount)

                        ++m_iPacketCounter;
                        sprintf(StatText, "Data Packet %d Received", m_iPacketCounter);
                        e_status->SetPaneText(1, StatText);

                        for(int iStatNum=1;iStatNum<=NUM_STATIONS;iStatNum++) {
                                try {
                                        // Deliver DataPacks to any waiting listener
                                        LoopObject[iStatNum-1]-
>deliverStation(LoopData[iStatNum-1]);
                                } // end try
                                catch (CORBA(SystemException) &sysEx) {
                                        cerr << "Unexpected system exception, exiting" << endl;
                                        cerr << &sysEx;
                                        exit(1);
                                } // end catch
                                catch(...) {
                                        cerr << "Unexpected exception inside ReceiveMsg" << endl;
                                        exit(1);
                                }  // end catch
                        }
                        try {
                                // Deliver DataPacks to any waiting listener
                                LoopObjectAll->deliverAllStations(LoopData);
                        } // end try
                        catch (CORBA(SystemException) &sysEx) {
                                cerr << "Unexpected system exception, exiting" << endl;
                                cerr << &sysEx;
                                exit(1);
                        } // end catch
```

```
                    catch(...) {
                            cerr << "Unexpected exception inside ReceiveMsg" << endl;
                            exit(1);
                    } // end catch

                    sprintf(StatText, "Station Packet %d", m_iPacketCounter);
                    e_status->SetPaneText(1, StatText);

            } // end if (DataTotal == (m_lPacketSize-1))

            if (DataTotal > m_lPacketSize) {
                    DataTotal = 0;
                    m_iDataRead = 0;
                    sprintf(StatText, "Corrupt Station Packet Received");
                    e_status->SetPaneText(1, StatText);
            } // end if (DataTotal > pSize)

    } // end else if (!FIRST_MSG)

} // end ReceiveMsg()


//////////////////////////////////////////////////////////////////////////
// OrbixTalk Functions

BOOL CLoopServDoc::RegisterAllTalkers()
{

    char LoopName[40];

    for(int iStatNum=1;iStatNum<=NUM_STATIONS;iStatNum++) {

            try {

                    // Set the topic name as the detector id
                    sprintf(LoopName,"otrmp//loop/station/30second/id%d",iStatNum);

                    // For some reason, this is causing all kinds of problems with the GUI...
                    // Register the talker objects
                    LoopObject[iStatNum-1] = myRegisterTalker (LoopName);

            } // end try
            catch (CORBA(SystemException) &sysEx) {
                    cerr << "Unexpected system exception, exiting" << endl;
                    cerr << &sysEx;
                    exit(1);
            } // end catch
            catch(...) {
                    cerr << "Unexpected exception inside RegisterAllTalkers" << endl;
                    exit(1);
            } // end catch

    } //end for

    try {
            // Set the topic name to ALL
```

```
                sprintf(LoopName,"otrmp//loop/station/30second/ALL");

                // Register the talker objects
                LoopObjectAll = myRegisterTalker (LoopName);
        } // end try
        catch (CORBA(SystemException) &sysEx) {
                cerr << "Unexpected system exception, exiting" << endl;
                cerr << &sysEx;
                exit(1);
        } // end catch
        catch(...) {
                cerr << "Unexpected exception inside RegisterAllTalkers" << endl;
                exit(1);
        } // end catch

        return TRUE;
}

// utility routine to register a Talker
// Note: This should be called from within a try/catch block
RDD_ptr CLoopServDoc::myRegisterTalker (const char* sLoop)
{
        // Get a CORBA::Object proxy built for us
        //
        // Note: Becase no Acknowledgement object is specified
        // when the talker is registered, this means that
        // invocations are "synchronous".
        // Such synchronous invocations do not return until
        // they are acknowledged as received by the message store.
        //
        CORBA(Object_ptr) obj = otalk->registerTalker ((const char *)sLoop, RDD_IR);

        // Narrow it into a LoopSensor
        return (RDD::_narrow (obj));

}
```

D.2.16 LoopServFrame.cpp

```
// LoopServFrame.cpp : implementation of the CLoopServFrame class
//
/////////////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "LoopServ.h"
#include "LoopServDoc.h"
#include "LoopServFrame.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CLoopServFrame
```

```
IMPLEMENT_DYNCREATE(CLoopServFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CLoopServFrame, CFrameWnd)
        //{{AFX_MSG_MAP(CLoopServFrame)
        ON_WM_CREATE()
        ON_COMMAND(IDM_FILE_CONNECT, OnFileConnect)
        ON_UPDATE_COMMAND_UI(IDM_FILE_CONNECT, OnUpdateFileConnect)
        ON_COMMAND(IDM_REGISTER_TALKERS, OnRegisterTalkers)
        ON_UPDATE_COMMAND_UI(IDM_REGISTER_TALKERS, OnUpdateRegisterTalkers)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
        ID_SEPARATOR,           // status line indicator
        ID_SEPARATOR,           // status line indicator
};

//////////////////////////////////////////////////////////////////////////
// CLoopServFrame construction/destruction

CLoopServFrame::CLoopServFrame()
        :m_bREGISTERED(FALSE), m_bCONNECTED(FALSE)
{
        // TODO: add member initialization code here

}

CLoopServFrame::~CLoopServFrame()
{
}

int CLoopServFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
                return -1;

        if (!m_wndStatusBar.Create(this) ||
                !m_wndStatusBar.SetIndicators(indicators,
                  sizeof(indicators)/sizeof(UINT)))
        {
                TRACE0("Failed to create status bar\n");
                return -1;      // fail to create
        }

        m_wndStatusBar.SetPaneInfo(0,0,SBPS_STRETCH,0);  // Make 1st Pane Beveled
   m_wndStatusBar.SetPaneInfo(1,NULL,NULL,105);

   char StatText[20];
   sprintf(StatText, "Not Connected");

   e_status = &m_wndStatusBar;
   e_status->SetPaneText(1, StatText);

        return 0;
}
```

```
BOOL CLoopServFrame::PreCreateWindow(CREATESTRUCT& cs)
{
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs

        return CFrameWnd::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CLoopServFrame diagnostics

#ifdef _DEBUG
void CLoopServFrame::AssertValid() const
{
        CFrameWnd::AssertValid();
}

void CLoopServFrame::Dump(CDumpContext& dc) const
{
        CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CLoopServFrame message handlers

//////////////////
// OnFileConnect() called when Connect option is chosen under File menu
void CLoopServFrame::OnFileConnect()
{
        CLoopServDoc* pDoc;
        char StatText[20];

        pDoc = (CLoopServDoc*) GetActiveDocument();

        if (pDoc->ConnectSocket()) {
                // connection to TMC data server was made successfully
//              pDoc->m_pCommandPacket->dataReq = NPIPE_5M;
                pDoc->m_pCommandPacket->dataReq = NPIPE_30S;
                pDoc->m_pCommandPacket->usCmd = CMD_NEWCON;
//              sprintf(pDoc->CommandPacket-
>userName,"%s",ProgramInfo.userName.GetBuffer(25));
//              sprintf(pDoc->CommandPacket-
>userLocation,"%s",ProgramInfo.userLocation.GetBuffer(25));
//              sprintf(pDoc->CommandPacket-
>userPhone,"%s",ProgramInfo.userPhone.GetBuffer(20));
//              sprintf(pDoc->CommandPacket-
>userEmail,"%s",ProgramInfo.userEmail.GetBuffer(40));
//              sprintf(pDoc->CommandPacket->userApp,"Spim");
                sprintf(pDoc->m_pCommandPacket->userName,"Greg");
                sprintf(pDoc->m_pCommandPacket->userLocation,"WE");
                sprintf(pDoc->m_pCommandPacket->userPhone,"22");
                sprintf(pDoc->m_pCommandPacket->userEmail,"22");
                sprintf(pDoc->m_pCommandPacket->userApp,"Jim");
```

```
                pDoc->SendMsg();

                sprintf(StatText, "Connected to Server");
          e_status->SetPaneText(1, StatText);

                m_bCONNECTED = TRUE;
                pDoc->m_bCONNECTED = TRUE;
        }
        else {
                // connection to TMC data server was not made successfully
                sprintf(StatText, "Server Not Found");

                e_status = &m_wndStatusBar;
                e_status->SetPaneText(1, StatText);
        }

}

void CLoopServFrame::OnUpdateFileConnect(CCmdUI* pCmdUI)
{
        pCmdUI->Enable(!m_bCONNECTED);
}

// OnFileConnect() called when Connect option is chosen under File menu
void CLoopServFrame::OnRegisterTalkers()
{
        CLoopServDoc* pDoc;
        char StatText[20];

        pDoc = (CLoopServDoc*) GetActiveDocument();

        // register the talker objects
        pDoc->RegisterAllTalkers();  // This is driving me nuts!!!!!

        sprintf(StatText, "Talkers Registered");
  e_status->SetPaneText(1, StatText);

        m_bREGISTERED = TRUE;

}

void CLoopServFrame::OnUpdateRegisterTalkers(CCmdUI* pCmdUI)
{
        pCmdUI->Enable(!m_bREGISTERED);
}
```

D.2.17 LoopServView.cpp

```
// LoopServView.cpp : implementation of the CLoopServView class
//
/////////////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "LoopServ.h"
#include "LoopServDoc.h"
#include "LoopServView.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//Thread for registering talkers
CWinThread *pThread;

/////////////////////////////////////////////////////////////////////////////
// CLoopServView

IMPLEMENT_DYNCREATE(CLoopServView, CView)

BEGIN_MESSAGE_MAP(CLoopServView, CView)
        //{{AFX_MSG_MAP(CLoopServView)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CLoopServView construction/destruction

CLoopServView::CLoopServView()
        //: CView(CLoopServView::IDD)
{
        //{{AFX_DATA_INIT(CLoopServView)
                // NOTE: the ClassWizard will add member initialization here
        //}}AFX_DATA_INIT
}

CLoopServView::~CLoopServView()
{
}

void CLoopServView::DoDataExchange(CDataExchange* pDX)
{
        CView::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CLoopServView)
                // NOTE: the ClassWizard will add DDX and DDV calls here
        //}}AFX_DATA_MAP
}

BOOL CLoopServView::PreCreateWindow(CREATESTRUCT& cs)
{
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs
        return CView::PreCreateWindow(cs);
}

void CLoopServView::OnDraw(CDC* pDC)
{
        CLoopServDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
}
```

```
void CLoopServView::OnInitialUpdate()
{
                CView::OnInitialUpdate();
}

/////////////////////////////////////////////////////////////////////////////
// CLoopServView diagnostics

#ifdef _DEBUG
void CLoopServView::AssertValid() const
{
        CView::AssertValid();
}

void CLoopServView::Dump(CDumpContext& dc) const
{
        CView::Dump(dc);
}

CLoopServDoc* CLoopServView::GetDocument() // non-debug version is inline
{
        ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CLoopServDoc)));
        return (CLoopServDoc*)m_pDocument;
}
#endif //_DEBUG
```

Appendix E - RDD Client Source Code (30-second)

E.1 CORBA Files

E.1.1 RDD.idl

Same as server code. See Appendix D.

E.1.2 RDD.hh

Created by the Orbix CORBA compiler, not included here.

E.1.3 RDDC.cpp

Created by the Orbix CORBA compiler, not included here.

E.1.4 RDDS.cpp

Created by the Orbix CORBA compiler, not included here.

E.2 30-second Loop Client Files

E.2.1 Const.h

```
// const.h

#define USE_DETECTORS        0
#define USE_STATIONS 1

#define NUM_STATIONS         864
#define NUM_DETECTORS        3500
#define MAX_IDS          NUM_DETECTORS

#define NUM_STATION_ATTRIBUTES   4
#define NUM_DETECTOR_ATTRIBUTES6
#define MAX_ATTRIBUTES       NUM_DETECTOR_ATTRIBUTES

#define DETECTOR_ID 0
#define DETECTOR_TIME 1
#define DETECTOR_VOLUME 2
#define DETECTOR_OCCUPANCY 3
#define DETECTOR_STATUS 4
#define DETECTOR_VALIDITY 5

#define STATION_ID 0
#define STATION_VOLUME 1
#define STATION_OCCUPANCY 2
#define STATION_STATUS 3
```

E.2.2 RDD_i.h

```
#ifndef rdd_ih
#define rdd_ih

#include "rdd.hh"

class RDD_i: public virtual RDDBOAImpl  {
public:
```

```
    virtual void deliverDetector (const Detector& package, CORBA::Environment
&IT_env=CORBA::default_environment) ;
    virtual void deliverStation (const Station& package, CORBA::Environment
&IT_env=CORBA::default_environment) ;
    virtual void deliverRampMeter (const RampMeter& package, CORBA::Environment
&IT_env=CORBA::default_environment) ;
    virtual void deliverAllDetectors (const AllDetectors package, CORBA::Environment
&IT_env=CORBA::default_environment) ;
    virtual void deliverAllStations (const AllStations package, CORBA::Environment
&IT_env=CORBA::default_environment) ;
    virtual void deliverAllRampMeters (const AllRampMeters package, CORBA::Environment
&IT_env=CORBA::default_environment) ;

        // constructor
        RDD_i (const char* RDDName):
                RDDBOAImpl (RDDName) {}  // Empty

        // destructor
        ~RDD_i () {}
};

#endif
```

E.2.3 stdafx.h (for precompiling)

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
//#include <afxdb.h>                    // MFC ODBC database classes

#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC support for Windows 95 Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT

#include <iostream.h>
#include <stdlib.h>

#include <orbixTalk.h>          // OrbixTalk definitions
```

E.2.4 RDD_i.cpp

```
// rdd_i.cpp : implementation file
//

#include "stdafx.h"
#include "rdd_i.h"
#include "const.h"

// Deliver functions, where code is added to instruct program what to do with the data when it arrives.
void RDD_i:: deliverDetector (const Detector& package, CORBA::Environment &IT_env) {
```

```
        cout << "Detector "<< package.id << "\ttime: " << package.time
                << "\tvol: " << package.volume << "\tocc: " << package.occupancy
                << "\tstat: "  << package.status << "\tval: " << package.validity
                << endl;

}

void RDD_i:: deliverStation (const Station& package, CORBA::Environment &IT_env) {
        cout << "Station "<< package.id << "\tvol: " << package.volume
                << "\tocc: " << package.occupancy << "\tstat: "  << package.status << endl;
}

void RDD_i:: deliverRampMeter (const RampMeter& package, CORBA::Environment &IT_env) {
}

void RDD_i:: deliverAllDetectors (const AllDetectors package, CORBA::Environment &IT_env) {
        cout << "Detector: " << _marker () << " now at " << package[0].volume << endl;
}

void RDD_i:: deliverAllStations (const AllStations package, CORBA::Environment &IT_env) {
}

void RDD_i:: deliverAllRampMeters (const AllRampMeters package, CORBA::Environment &IT_env) {
}
```

E.2.5 Stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
//        LoopServ.pch will be the pre-compiled header
//        stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

E.2.6 Loop_listener.cpp

```
////////////////////////////////////////////////////////////////////
// loop_listener.cpp
////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "rdd.hh"
#include "rdd_i.h"
#include "const.h"

int main (int argc, char * argv[])
{
        OrbixTalk_ptr otalk = NULL;
        RDD_ptr testLoop[NUM_STATIONS];
        char LoopName[50];

        // initialize the talkers
        for(int statNum=1; statNum <= NUM_STATIONS; statNum++) {
                testLoop[statNum-1] = NULL;
        }

        cout << endl << "OrbixTalk API Loop Sensor listener :";
        cout << "(Uses the OrbixTalk RMP protocol)" << endl << endl << endl;
```

```
try {
        //      Initialise the OrbixTalk Manager object
        otalk = OrbixTalk::initialise ();

        // Build the Listener objects
        //for(int iStatNum=1; iStatNum <= NUM_STATIONS; iStatNum++) {
        for(int iStatNum=1; iStatNum <= 100; iStatNum++) {
                sprintf(LoopName,"otrmp//loop/station/30second/id%d",iStatNum);
                testLoop[iStatNum-1] = new RDD_i(LoopName);
        }

        // Register the listeners
        //for(iStatNum=1; iStatNum <= NUM_STATIONS; iStatNum++) {
        for(iStatNum=1; iStatNum <= 100; iStatNum++) {
                otalk->registerListener (testLoop[iStatNum-1]);
        }

        // Need to execute the event loop
        // to process incoming and outgoing events
        CORBA(Orbix).processEvents (CORBA(Orbix).INFINITE_TIMEOUT);

} catch (CORBA(SystemException) &sysEx) {
        cerr << "Unexpected system exception" << endl;
        cerr << &sysEx;
        exit(1);
} catch(...) {
        cerr << "Unexpected exception " << endl;
        exit(1);
}

// Release memory associated with _ptr types
for(int iStatNum=1; iStatNum <= NUM_STATIONS; iStatNum++) {
        CORBA(release)(testLoop[iStatNum-1]);
}


cout << "--- Listener end..." << endl;
return (0);
}
```

Appendix F - ScopeServer Details

F.1 Installation Procedure

Disks needed:

- ScopeServer32 (two disks)
- Interface Developer's Kit (two disks)
- Traffic Alarm monitor (two disks)
- "Special" disk

Steps:

1. Insert the ScopeServer Interface Developer's Kit disk 1

   Run "a:\Setup.exe" from the disk

   Select "ScopeServer-32"

   Set the directory to which the software will be stored (like C:\ASCOPE)

   Name the Program Manager Group "ScopeServer"

2. Insert ScopeServer-32 disk 1

   Set the directory path to what you chose earlier (C:\ASCOPE)

   Insert ScopeServer-32 disk 2

3. Insert the ScopeServer Interface Developer's Kit disk 2

   Install OmegaSim Autoscope Simulator from DOS prompt by typing

   "INSTALL A: C:\ASCOPE"

4. Insert Traffic Alarm Monitor disk 1

   Run "a:\Setup.exe" from the disk

   Check "TCP" and enter the computer host name (like Jupiter)

   Install the software in the same directory as before (C:\ASCOPE)

Insert Traffic Alarm Monitor disk 2

Set the Program Manager Group to "ScopeServer"

5. Insert "Special" disk

   Copy "arch96.exe" file to C:\ASCOPE\WIN32

   Copy entire "TAM" directory to C:\ASCOPE

F.2 Running the ScopeServer and the Traffic Data Archiver Applications

The ScopeServer is used to communicate with multiple machine-vision processors (MVPs) simultaneously. It will collect, process, and distribute traffic data to client applications that require it, acting as a data broker. There are two example ScopeServer client applications that you can install:

- Traffic Data Archiver – allows the storage of request and poll data from MVPs
- Traffic Alarm Monitor – provides real-time traffic alarm information to human operators

Before running the Traffic Data Archiver application, verify the following:

- Autoscope units are functioning properly with good video signals.
- Each Autoscope unit has accurate detector files that reflect the video signal being received. Note that you must have detectors grouped into "stations" in your detector files in order to collect data from them using ScopeServer.
- Both A/B switches are set to B.

Steps:

1. Start the ScopeServer by opening Programs => ScopeServer => ScopeServer for Windows32. This monitors all communications with the MVPs and displays

messages to the screen. Minimize the ScopeServer window to avoid cluttering the desktop.

2. Start the Traffic Data Archiver by opening Programs => ScopeServer => Traffic Data Archiver TCP-IP.
3. Set the location where the data files will be stored by choosing "Set Data Directory" under the File menu option.
4. Open communications with the MVPs by choosing "Open Communications [Shared]" under the Comm menu option. (Note that there are two options: Shared and Exclusive. Use Shared for this application.)
5. Select the MVP with which to communicate. (You can select all three at once. To avoid confusion, it is recommended you select one at a time.)
6. Set the polling interval by choosing "Select Poll intervals" under the Data menu option. When you are finished select OK. The application will begin collecting data and saving it in a file in the directory you specified earlier. Data is also displayed on the screen as it is collected.

Appendix G – Project Summary Presentation

The following Presentation was given to University and MnDOT officials on January 27th, 1998. We include it in this report because of the good overview it gives on the research program were this phase was part of.

I-394 Laboratory Phase II Results

January 27, 1998

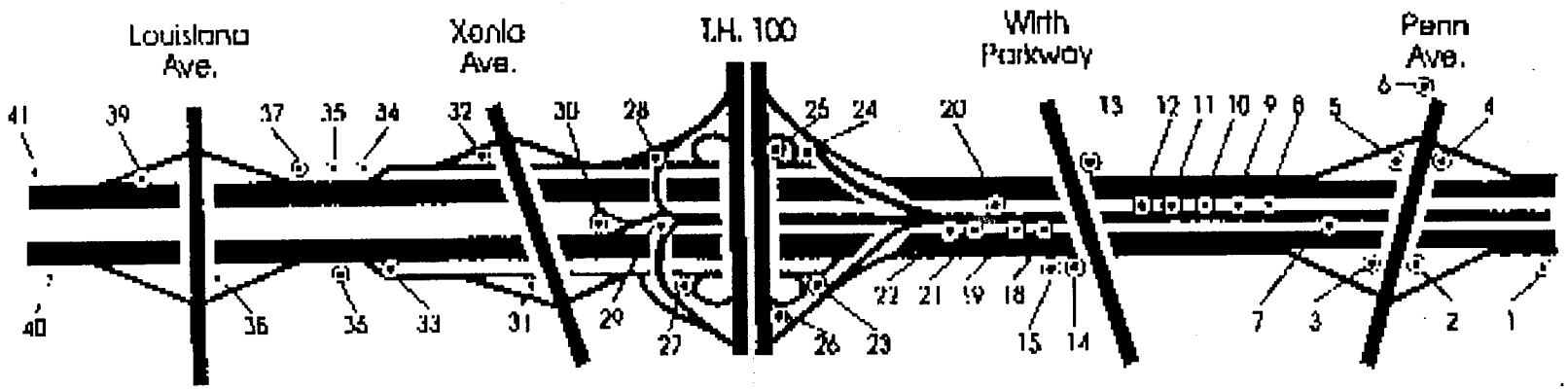
Overview

- I-394 Lab Objective
- Historical Perspective
- Project Phases
- Phase I Summary
- Phase II
  - Objectives
  - Actions
  - Results
- Phase III Plans

I-394 Lab Objective

- To develop advanced traffic sensor data collection, management, and distribution systems which provide accurate, flexible, and wide-area traffic information
  - for deployment in the next-generation traffic management center, and
  - for enabling advanced research as part of the ITS Laboratory infrastructure

Historical Perspective: The I-394 Machine-Vision System (MVS)



- MVS deployment completed in 1993
- 36 cameras over 7 miles, 9 MV Devices
- TMC controllable
- Never utilized effectively

Project Phases

- Phase I: I-394 MVS analysis and I-394 Lab activation
- Phase II: Traffic data management system design and development
- Phase III: Integration of I-394 Lab, Simulation Lab, and TRACLAB

Phase I Objective

- To develop a laboratory environment that uses machine-vision to collect and manage accurate, detailed, and continuous traffic data over long stretches of freeway for use in the design, development, calibration, testing, and evaluation of traffic models, traffic management systems, and other ITS applications

Phase I Plan

- To "resurrect" the dormant I-394 MVS and activate the I-394 Lab by establishing communications to and control over the MVS from the ITS Laboratory
- I-394 Lab control goals
  - Video
  - Detector Placement and Types
  - Data Collection

Phase I Issues

- MVS developed in two distinct stages, resulting in a poorly designed and difficult-to-use system
- MVS poorly documented
- MVS never maintained
- Political/jurisdiction issues

Phase I Results

- Complete analysis/documentation of the MVS specifications
- MVS design modifications
- MVS diagnostics and repair negotiations
- Established communications between the ITS Lab and the I-394 MVS
  - Video
  - Serial

Phase I Conclusions

- Existing I-394 MVS needed significant repair work (especially communications) to be a useful backbone for the I-394 Lab
- Project work could not wait for nor rely on these repairs to happen in a timely manner
- Future work would concentrate on aspects of traffic sensor data systems that we have more control over

Phase II Objectives

- To develop a Machine-Vision Lab for training and research using the available I-394 camera feeds and local ITS Lab machine-vision processors
- To design and develop a Traffic Data Management System for loop-sensor data collection, management, distribution, and storage

Phase II Actions

- Design and implement Machine-Vision Lab
- Design Traffic Data Management System
- Review CTS & Mn/DOT related database projects/work
- Implement most necessary portion(s) of the design not covered by other project work

Machine-Vision Lab Features

- Live video feeds from I-394 and local cameras
- Three local machine-vision devices
- Machine-vision training
- Machine-vision research
- Centralized data collection from one or all machine-vision devices (up to 6 cameras)

Machine-Vision Lab Design