Integrating Clarus Data in Traffic Signal System Operation

A Survivable Real-Time Weather-Responsive System

www.its.dot.gov/index.htm
Final Report — November, 15, 2011
FHWA-JPO-12-016

U.S. Department of Transportation
Federal Highway Administration
Research and Innovative Technology Administration

Produced by Produced by the University of Idaho
for:
ITS Joint Program Office
Research and Innovative Technology Administration
U.S. Department of Transportation

Notice

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA-JPO-12-016		

4. Title and Subtitle	5. Report Date
Integrating Clarus data in traffic signal System Operation: A Survivable real-time weather-responsive system	**07 11 2011**
	6. Performing Organization Code KLK818

7. Author(s)	8. Performing Organization Report No.
Ahmed Abdel-Rahim, Axel Krings, and Michael Dixon	NIATT-11-08

9. Performing Organization Name And Address	10. Work Unit No. (TRAIS)
University of Idaho P.O. Box 440901 Moscow, Idaho, 83844-0901	**11. Contract or Grant No.** BAA No. DTFH61-10-P-00123

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Federal Highway Administration Office of Transportation Operations 1200 New Jersey Ave, S.E. , E86-302 Washington, D.C., 20590	Final report. September 24, 2010 to December 23, 2011 **14. Sponsoring Agency Code**

15. Supplementary Notes
The Contracting Officer Technical Representative (COTR) for the project is:
C. Y. David Yang, Ph.D., Office of Operations R&D , Turner-Fairbank Highway Research Center, FHWA, U.S. DOT
6300 Georgetown Pike, McLean, VA 22101, (202) 493-3284 , (202) 493-3419, Fax

16. Abstract

This report presents a prototype of a secure, dependable, real-time weather-responsive traffic signal system. The prototype executes two tasks: 1) accesses weather information that provides near real-time atmospheric and pavement surface condition observations and 2) adapts signal timing in response to inclement weather. The proposed system architecture employs two revolutionary software design approaches: 1) Design for Survivability and 2) software performance measurement at the task level. Furthermore, the software design incorporates self-diagnostic techniques for fault detection and recovery to maximize the survivability and the security of the system. Minimal hardware is required for full implementation of the system as it operates and achieves its potential using current traffic controller and cabinet standards and technologies. As a result, it is compatible with future applications within the FHWA's connected-vehicle framework. The weather-responsive traffic signal system presented in this report serves as a major milestone in the development of secure and dependable real-time traffic control system applications.

17. Key Words	18. Distribution Statement
Clarus, Responsive, Weather, Traffic Signal System, Survivable	Unrestricted; Document is available to public through the National Technical Information Service, Springfield, VT

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	**Unclassified**	**43**	**N/A**

Preface/ Acknowledgements

This research has been supported by grant DTFH61-10-P-00123 from the Federal Highway Administration – U. S. Department of Transportation. The authors would like to thank David Yang, the project's Contracting Officer Technical Representative (COTR), from the FHWA Office of Operations Research and Development, Turner-Fairbank Highway Research Center for his valuable contribution and guidance throughout the project duration. The authors would also like to thank Paul Pisano (team leader- FHWA's Road Weather Management Program) and Paul Olson (ITS technology Engineer-FHWA Resource Center) for serving in the project's technical advisory committee and Brenda Boyce (Mixon/Hill, Inc.) for facilitating access to the Clarus data. The authors would like to extend their appreciation to Victor Balogun, Saad Alshomrani, and Kristian Henrickson for their valued contribution in different project tasks.

Table of Contents

List of Tables

List of Figures

Executive Summary

This report presents a prototype of a reliable, secure, and survivable, real-time weather-responsive traffic signal system with the intent of improving the safety and efficiency of traffic signal system operations during inclement weather conditions. The prototype executes two tasks: 1) accesses weather information that provides near real-time atmospheric and pavement surface condition observations and 2) adapts signal timing in response to inclement weather. Development of the prototype followed a standard systems engineering process that included six steps: reviewing the resources, defining the system specifications, designing the system, creating the data interface and analyzing the data, developing the testing environment, and performing verification and timing analysis.

The prototype system architecture includes a microprocessor, external to the traffic controller, that receives Clarus data, analyzes the relevant data, and communicates necessary signal timing changes to the traffic controllers. Current technology supports the proposed system development. Microprocessor traffic controller NTCIP-based communications were tested verifying that the necessary read and write capabilities are available from the microprocessor to any NTCIP-compliant traffic controller.

The weather data was accessed through a subscription to the Clarus system web interface. Different observation types reported in the Clarus data system were used to determine air and surface temperature, roadway surface condition status, precipitation type and rate, and visibility level at or near the environmental sensing station. The availability and accuracy level of the weather data reported in the Clarus system provided reliable estimates of the weather, road surface condition, and visibility level needed for weather-responsive traffic signal system applications.

The survivable weather-responsive traffic signal system developed as part of this project was evaluated and tested by conducting two analyses: traffic system benefits analysis and software testing and risk analysis. The potential crash reduction benefits, expressed as the percent reduction in total, rear-end, and crossing conflicts, are highest during snowy and icy weather conditions. The potential crash reduction benefits increase as the traffic volume level increases. Rear-end conflicts are the conflict type projected to be most eliminated by a weather-responsive traffic signal system with a potential average reduction of approximately 22 percent for moderate volume levels and 43 percent for high volume levels. The weather-responsive signal timing plans also show considerable potential in reducing traffic delays and stops. Again, the percent reduction increases as the traffic volume level increases. The potential reduction in delays and stops seems consistent with what has been reported in the literature.

The software architecture of the proposed work, with its design for survivability approach, is a fundamental building block in a highly networked and interactive communications system. The overall system architecture was comprised of multiple components, the executing program, and the contingency management system. The sole purpose of the latter was to watch the execution in real-time and react to unwanted changes as they would occur as the result of system components malfunctioning or communication failure. Survivability measures during the design and operation of

the system were centered around the Operation Monitoring and Contingency Management System, which interfaced to the software system via the instrumentation telemetry. The adaptability and recovery from any unintended or maliciously induced operations/profiles was determined by the survivability policy and was handled by the Contingency Management System.

The weather-responsive system developed in this project has five innovations:

- The system operates and achieves its potential using current traffic controller and controller cabinet technologies.
- The system is compatible with future applications within the FHWA's connected-vehicle initiative.
- Minimal hardware, in addition to traffic controllers, is required for full system implementation.
- Computer driven algorithms implement traffic signal control decisions using Clarus data.
- The proposed system architecture employs two revolutionary software design approaches: design for survivability and software performance measurement at the task level.

Furthermore, the software design incorporates self-diagnostic techniques for fault detection and recovery to maximize the survivability and the security of the system. Because the proposed system has very similar computational requirements to other field traffic control applications, it serves as a major milestone in the development of secure and dependable real-time traffic control systems.

Future research should focus in three areas:

- Field testing the system at signalized intersections in a variety of weather conditions;
- Expanding control modifications to include other traffic control parameters, such as passage time, minimum green, and offsets;
- Increasing the power of the system to maintain reliable, secure, and survivable traffic signal service.

Chapter 1. Introduction

1.1 Overview

Adverse weather conditions such as rain, fog, and snow can reduce pavement friction and visibility, thereby impairing the ability of drivers to operate their vehicles safely. This reduces roadway capacity and significantly affects both the safety and efficiency of arterial system operations. The effect of weather on traffic crashes and highway safety is well documented in the literature. Ye et al. reported that weather-related crash fatalities account for 17 percent of all traffic fatalities each year [2009]. Several studies found that weather significantly increases crash risk [Pisano 2008 and Andrey 2005], with one study suggesting that snow increases crash risk by approximately 120, 80, 40, and 40 percent for minimal, minor, major, and fatal injuries, respectively [WTI 2009]. In terms of weather effect on the traffic operations along arterials, several studies found that traffic signal timing plans used under normal conditions became problematic under adverse weather. The reduction in average speeds and saturation flow rates, and the increase in start-up delays, make normal signal timing parameters unsuitable during inclement weather. In addition, with reduced pavement friction and visibility, default all-red and amber clearance intervals become unsafe as motorists are more likely to be trapped in dilemma zones at the onset of red. Several studies have investigated the effect of inclement weather on various signal timing traffic parameters [see Gillam 1992, Bernardin 1995, Perrin 2002, and Seli 2004]. Studies have shown that weather-responsive signal timing plans can improve both the safety and efficiency of the traffic signal system operations. Simulation studies revealed benefits of approximately 7 percent to 23 percent reduction in average delay, 4 percent to 9 percent reduction in vehicle stops, and 3 percent to 12 percent increase in average speeds [Pisano 2004 and Al-Kaisy 2006]. In addition, several signal timing plans were adjusted for inclement weather and deployed in the field [Bernardin 1995 and Ye 2009]. However, the adjusted signal timing plans in these studies were manually implemented by transportation system operators when conditions to trigger the timing plans were met.

The goal of this project is to develop a prototype of a real-time weather-responsive traffic signal control system with the intent to improve the efficiency and safety of traffic signal operations during inclement weather conditions. The system developed as part of this project is capable of receiving and analyzing road weather information from the Clarus weather data system and adapts signal timing in response to changes in road surface conditions and/or visibility level. The Clarus Initiative is a joint effort of the U.S. Department of Transportation Intelligent Transportation Systems (ITS) Joint Program Office and the Federal Highway Administration's (FHWA) Road Weather Management Program, which resides in the Office of Transportation Operations. Clarus (which is Latin for "clear") is an initiative to develop and demonstrate an integrated surface transportation weather observation data management system, and to establish a partnership to create a nationwide surface transportation weather observing and forecasting system (FHWA 2009). The Clarus System provides near real-time atmospheric and pavement observations from participating states' Environmental Sensor Stations (ESS). The FHWA's Clarus system functions include: data assimilation, quality checking, and data dissemination.

The weather-responsive system developed in this project has five innovations. First, the system operates and achieves its potential using current traffic controller and controller cabinet technologies.

Second, the system is compatible with future applications within the FHWA's connected-vehicle initiative. Third, minimal hardware, in addition to traffic controllers, is required for full system implementation. Fourth, computer driven algorithms implement traffic signal control decisions using Clarus data. Fifth, the proposed system architecture employs two revolutionary software design approaches: design for survivability and software performance measurement at the task level. Furthermore, the software design incorporates self-diagnostic techniques for fault detection and recovery to maximize the survivability and the security of the system.

1.2 Project Tasks

To accomplish the project's goal, the following six tasks were executed:

1. Review of Resources

   In the initial phase, available resources were assessed to examine and document methods to access and manipulate the Clarus data as well as traffic controller objects. It also included designing experiments to test the Clarus data access. To implement the concept of design for survivability, the effects of hardware constraints on software design were examined in greater detail to adopt approaches that ensure multiple software tasks can be completed in parallel with a full assessment of real-time feasibility with respect to hard and soft task deadlines.

2. Define System Specification

   This task has several subtasks in which the system specifications were derived: 1) hardware implementation, 2) functional objectives and integrity, and 3) software implementation according to the design for survivability philosophy. System specifications also included the operations performed by the operating monitoring engine, as well as descriptions of which traffic controller functionalities are monitored and the derivation of basic definitions of normal and abnormal operations. System specifications are also used to design the monitoring interface that is the basis for the contingency management system that specifies the different states of the system observed by the operation monitoring engine, which include fail-safe state, default operations, and operations in an elevated awareness state. Responses, such as recovery after failure and adaptability as the result of observed behavior, are also part of the contingency management system specifications. In addition to the software functional specifications, this task also included defining system testing requirements. This involves the signal timing analysis procedures and compliance with the requirements of the infrastructure used as part of the system.

3. System Design

   System design involves designing several fundamental components according to specifications: the system hardware, the interfacing technology, and the system operational software and contingency management system. Industry design processes were adopted to ensure that the system specifications are adhered to rigorously.

4. Data Interface and Analysis

   This task addressed the practical implications of Clarus data communications and data manipulation in the local processing units. It included the derivation of a scalable local Clarus client. Specifically, the overhead associated with communications and data queries needed to be analytically and experimentally established to achieve a scalable design. Because the algorithms and data processing share the same microprocessor, analysis of this overhead needed to explore trade-offs between control algorithm

complexity within the Algorithm Engine and the data processing requirements needed to utilize Clarus data. The software design strategy used in this project was “operation with imprecise results,” where the quality of the outcome increases with the amount of data and time available to the algorithms residing in the Algorithm Engine. This design strategy increases the possible options to deal with bottle-necks in the Clarus Data Management Engine, guaranteeing acceptable output.

5. Developing the testing environment

   A hardware-in-the-loop simulation (HILS) model was used to test and validate the real-time weather-responsive system developed as part of this project. The HILS model used in this project included a workstation running the VISSIM microscopic simulation model for the network, a traffic controller, a controller interface device (CID) to facilitate the exchange of data between the simulation model and the traffic controller, and an external processing unit that runs the software application and the weather-responsive control algorithm and is connected to both the Clarus system and the traffic controller.

6. Verification and Timing Analysis

   This task addressed standalone system testing and integrated systems testing and analysis. System verification in the context of design for survivability is more complicated than in a traditional approach that does not design the security and survivability considerations into the system. In this project, the system was studied in real time to observe its behavior in the presence of induced malicious acts and other faults. The interplay of functionality execution and adaptive control was studied to determine how the algorithm selection processes affect traffic signal timing and to measure the efficiency with which the controller monitoring functions execute.

1.3 Report Organization

This report is organized in five chapters. After the introduction, Chapter 2 provides an overview of the real-time weather-responsive system design and architecture. Chapter 3 documents the use of environmental data in responsive traffic signal control decisions. Chapter 4 documents the potential safety and operational benefits of weather-responsive traffic signal systems. Chapter 5 provides an overview of the software design architecture, development, and testing. Finally, chapter 6 includes conclusions and closing remarks.

Chapter 2. System Description and Communication Architecture

2.1 Overview

This chapter presents the architecture of a prototype for a secure, dependable, real-time weather-responsive system. Real-time control systems, especially those governing critical infrastructures such as transportation, need to be reliable and secure under normal operating conditions and survivable under abnormal conditions. Traffic control applications should be designed and operated so that essential services will survive even in the presence of component failure. Survivability, for the purpose of traffic control applications, is defined as the capability of a control system to fulfill its mission in a timely manner, even in the presence of a component or communication failure. The prototype system design incorporates state-of-the-art secure and dependable software design concepts to ensure accurate execution of two tasks. For the first task, the system accesses near real-time atmospheric, weather, visibility, and road surface condition information from the FHWA' Clarus data system. The second task adapts signal timing in response to inclement weather.

The proposed system architecture employs two revolutionary software design approaches: 1) Design for Survivability and 2) a Measurement-Based Methodology. The latter is for critical applications that rely on measurements of the operational system and dependability models to quantify reliability and system performance with certain user-defined confidence levels. Furthermore, the software design incorporates self-diagnostic techniques for fault detection and recovery to maximize the survivability and the security of the system. Minimal hardware is required for full implementation of the system as it operates and achieves its potential using current traffic controller and cabinet standards and technologies. As a result, it is compatible with future applications within the FHWA's connected-vehicle initiative. Because the proposed system has very similar requirements to other traffic control applications, it serves as a major milestone in the development of secure and dependable real-time traffic control systems.

2.2 Proposed Communication Architecture

The communication architecture of the proposed real-time weather-responsive traffic signal control system is shown in Figure 1. The system includes a microprocessor, external to the traffic controller, that receives Clarus data, analyzes the relevant data, and communicates necessary signal timing changes to the system operator for approval. Upon approval, signal timing changes are then made in the traffic controllers. Signal timing plan adaptations include changes such as modified all-red and yellow clearance intervals or traffic signal efficiency parameters such as minimum green, maximum green, or passage time, as well as different coordination parameters. Suggested changes depend on multiple factors such as approach speed, pavement surface conditions, visibility, and the mode of signal operations. Current technology supports the proposed system development, where microprocessor traffic controller communications were tested, verifying that the necessary read/write

capabilities are available from the microprocessor to the controller [Ahmed 2010]. In addition, recent advances in software design make fault detection and recovery possible for real-time in-field control applications. For this prototype, the Rabbit 5000 microprocessor fulfills the role of the local processing unit shown in Figure 1.

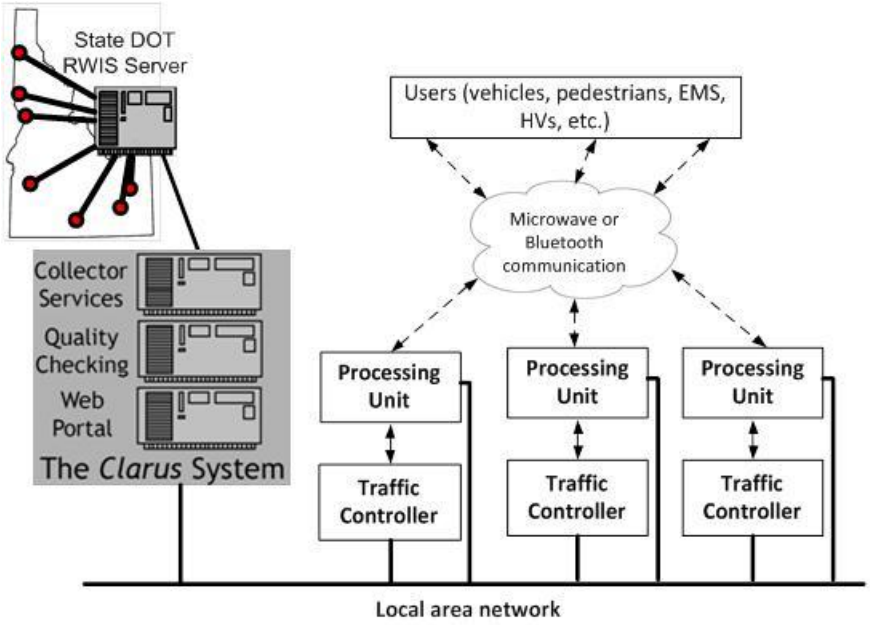


Figure 1. Communication Architecture for Clarus Integration into Traffic Signal System

The Rabbit microprocessor is the core hardware in the system that communicates with the traffic controller through the Ethernet. To facilitate communications, the controller and microprocessor must follow the National Transportation Communications for ITS Protocol (NTCIP) communication standard [AASHTO 2005], a family of standards for transmitting data and messages between different devices used in ITS application. The Dynamic Object STMP/UDP/IP Ethernet protocol stack is used to facilitate the NTCIP-based communication between the microprocessor and the traffic controller. A computer, connected to the microprocessor through the cabinet serial connection, is used to set up and add the control logic to the microprocessor. Because the microprocessor is directly connected to the traffic controller through the Ethernet port, the connection is not sensitive to the cabinet configuration. However, the microprocessor requires an additional 110 volt power connection. This connection method should be possible in any NTCIP compliant controller.

The Rabbit 5000 microprocessor meets the functional requirements for real-time traffic control feedback. This type of microprocessor is designed specifically for embedded control, communications, and Ethernet connectivity. In addition, the microprocessor's processing speed is 55.5 MHz clock speed, which is more than adequate for traffic control applications. It also features a battery-operated real-time clock. In addition to the Ethernet port, it offers a 20-bit address bus, 8-bit data bus, and 3 chip select lines. Two output-enabled lines, and 2 write-enabled lines can be directly interfaced with up to 8 Flash/SRAM devices.

2.3 Microprocessor Communication with Traffic Controllers

To implement the real-time weather-responsive control algorithm, the microprocessor used in the system reads weather data from the Clarus weather data system and also the signal status, phase timing plan, next phase, and phase omit data from the traffic controller. If the control algorithm determines that a certain change to the signal timing plan is needed, the Rabbit microprocessor disseminates the adaptive feedback control decision to the traffic controller. This read/write data exchange between the microprocessor and the traffic controller is facilitated through NTCIP standards.

NTCIP defines a collection of standards-based communication protocols and data profiles used in the transportation industry for center-roadside, center-center, and vehicle-roadside communications. NTCIP-based software and hardware devices can help achieve interoperability and interchangeability. NTCIP 1202 – Actuated Signal Controllers (ASC) [AASHTO 2005] defines an open and standard communications protocol for data exchange between software applications and traffic signal hardware. It defines elements for controlling, managing, and monitoring actuated traffic signal controller units including phases, rings and sequence; detectors; special functions; coordination; time base control; preemption; overlaps; and upload and download. NTCIP 1202 uses two communication protocols. The first is the Simple Network Management Protocol (SNMP), which defines rules for reading/writing objects to the controller (Get, Set, Get-Next, Trap) and a mechanism for status reporting, control upload/download and time broadcast. The second is the Simple Transportation Management Protocol (STMP), which provides bandwidth and processing efficiency alternate to SNMP for status reporting of dynamic objects to concatenate objects. Table 1 shows the data accessibility of different objects used in this study within an NTCIP compliant traffic controller. The full list of NTCIP objects and their accessibility status are provided in Appendix B.

Table 1. Example of Accessibility Status of NTCIP 1202 Objects

Object Name	Accessibility	Object Name	Accessibility
phaseStatusGroupGreens	Read-only	phaseMaximum	Read-write
phaseStatusGroupYellows	Read-only	phaseStatusGroupPhaseNexts	Read-only
phaseStatusGroupReds	Read-only	phaseControlGroupPhaseOmit	Read-write
phaseMinimumGreen	Read-write	Sensor (1), (2), and (3) input	Read-only
phasePassage	Read-write		

It should be noted that all NTCIP dynamic objects related to phase status are read-only. As a result, termination of a phase is not a feasible control feedback option. Instead, phase operations must be influenced by changing parameter values such as min green, max green, passage time, etc. and this was the approach taken in this project. A program, installed on the Rabbit, governs its operations and is programmed using the Dynamic C® software development system. Dynamic C is an integrated C compiler, editor, loader, and debugger fashioned for the Rabbit microprocessor. There are two basic sections in the code. The first section is developed for communications, sending data requests, receiving data, and sending control feedback to the operator and to the controller. The second section is written to process data, determine control decisions, and send control feedback.

Chapter 3. Environmental Data Use

3.1 ESS Observation Types

The objective of this part of the analysis is to aid in the design of decision trees that use the Clarus data to determine, with the highest possible degree of reliability, the weather conditions, the visibility level, and the roadway surface conditions at or near the ESS location. These decision trees are part of the algorithm used to determine the weather-responsive traffic signal control decisions. Before deciding on which ESS parameters should be used in these decision trees, it was important to determine which Clarus observation types are typically available and produce an estimate of the availability of each observation type. The availability of each observation type is quantified as a ratio of the number of Road Weather Information System (RWIS) stations nationwide that report this specific observation type. Several blocks of archived Clarus weather data were surveyed as part of this step of the analysis. It should be noted that the observation availability is not a measure of the accuracy or the reliability of the data; instead, it is just an indicator of which observations are more common in the Clarus data. The full list of ESS observation types are listed in Table 2. The observation types are categorized into five categories based on their availability in the archived Clarus data sets. The five categories included are: very common, common, less frequent, infrequent, and very infrequent. Some ESSs are equipped with logical sensors that report the weather "situation" by combining various physical observations (precipitation type/rate, temperature, etc.), but a greater number of ESSs simply report individually measured weather parameters. Logical sensor observations are identifiable as those with a "situation" suffix in Table 2.

3.2 Estimated Accuracy of ESS Observation Types

Previous research indicates that signalized intersections' traffic-flow parameters, such as saturation flow, free flow speed, and start-up lost time, are most sensitive to changes in roadway surface conditions, precipitation type and rate, and visibility level. A number of ESS observation types can be used to describe the road surface and visibility conditions. To ascertain observation type reliability, observations considered relevant to this purpose were selected from three RWIS stations near the city of Moscow, Idaho, and compared to historical weather data in the area. These comparisons were made on a series of days for which weather conditions were known in the vicinity of Moscow, ID during the winter/spring of 2010-2011. These observations along with their respective estimated accuracy are shown in Table 3. The observations were categorized into four groups based on their accuracy level: very accurate (90 percent or more accuracy), accurate (80 percent to 89 percent accuracy), likely accurate but difficult to verify, and accuracy not known as data is not available.

Table 2. Occurrence of Observation Types in Archived Clarus Data

Primary/Very Common		
WindSensorAvgSpeed WindSensorSituation windSensorGustSpeed essAirTemperature windSensorAvgDirection	essRelativeHumidity essDewPointTemp essSurfaceTemperature essSubSurfaceTemperature essSurfaceStatus	
Secondary/Common		
PrecipType Precipintensity essPrecipRate essVisibility essAtmosphericPressure		
Less Frequent		
essSurfaceSalinity essSurfaceFreezePoint essSurfaceIceOrWaterDepth essPrecipitationOneHour essPrecipitationThreeHours essPrecipitationSixHours	essPrecipitationTwelveHours essPrecipitation24Hours essPrecipSituation essPrecipYesNo essMaxTemp essMinTemp	windSensorSpotDirection windSensorSpotSpeed essWetBulbTemp essVisibilitySituation essPavementTemperature
Infrequent		
essAdjacentSnowDepth essSnowfallAccumrate essPrecipitationStartTime essPrecipitationEndTime essTotalRadiation	essSurfaceBlackIceSignal essPavementSensorError essSurfaceConductivityV2 IcePrecent precip10min	
Missing or Very Infrequent		
essSubSurfaceMoisture essSubsurfaceSensorError pavementSensorTemperatureDepth essTotalRadiationPeriod essTotalSun	essCloudSituation essInstantateousTerrestrialRadiation essInstantateousSolarRadiation waterLevelSensorReading essRoadwaySnowDepth	essRoadwaySnowpackDepth essIceThickness

Table 3. Clarus Observations Related to Traffic Signal Timing Parameters

Clarus Observation	Accuracy Level
essAirTemperature	VA[1]
essSurfaceTemperature	LA[2]
essSubSurfaceTemperature	NA[3]
essSurfaceStatus	VA
PrecipType	A[4]
Precipintensity	A
essPrecipRate	A
essVisibility	A
essSurfaceSalinity	NA
essSurfaceFreezePoint	NA
essSurfaceIceOrWaterDepth	NA
essPrecipSituation	A
essPrecipYesNo	VA
essPavementTemperature	NA
essSurfaceBlackIceSignal	NA
essRoadwaySnowDepth	NA
essRoadwaySnowpackDepth	NA
essIceThickness	NA

[1]VA = very accurate (>90%), [2]LA = likely accurate but difficult to verify, [3]NA = not known/not available, [4]A = accurate (80% - 90%)

3.3 ESS Observation Types Relevant to Traffic Signal System Operations

Table 4 shows the ESS observation types that can be used to determine surface/air temperature and precipitation conditions at the site location. The column labeled "Redundant" highlights observation types that are redundant or analogous to the observation types selected for this project. The redundant observation types will be used only if the main observation types are not available or determined to be unreliable for use in the analysis. Observation type "*essVisibility"* is the only observation in Table 3 that describes visibility with an estimate of a visibility level (distance). The other visibility related observation type "*essVisibilitySituation"* (Table 2) only reports the type of visibility impediment such as fog, dust, etc. Therefore, visibility distance level determination is based on the "*essVisibility"* observation type.

Observation types "*essSurfaceIceOrWaterDepth"* and "*essSurfaceBlackIceSignal*" were considered as possible indicators of surface ice, but it was decided that a combination of "*essSurfaceStatus,*" and "*essSurfaceTemperature*" would utilize a more intuitive combination of data elements and provide the same information. In addition, "*essSurfaceIceOrWaterDepth*" and "*essSurfaceBlackIceSignal*" are both reported infrequently and are unavailable in many areas. Table 5 lists the Clarus observation types that were selected for use in the decision trees that were used in this project. The observation types were selected based on their availability, estimated accuracy, and relevancy to air temperature, surface status, precipitation type and amount, and visibility level. These data elements are among the

most available data reported by different sensor stations, and are sufficient and accurate enough to determine weather and road conditions that are relevant to traffic signal system operations.

Table 4. Clarus Observations Used to Determine Current Temperature and Precipitation

Weather Element	Observations Used	Redundant Observations
Temperature	essAirTemperature essSurfaceTemperature	essSubSurfaceTemperature essPavementTemperature
Precipitation	essPrecipRate PrecipType essPrecipYesNo	Precipintensity essPrecipSituation

Table 5. Clarus Observation Types Selected for Use in Decision Structure

Weather Element	Selected Observations Type
Temperature	essAirTemperature essSurfaceTemperature
Surface	essSurfaceStatus
Precipitation	essPrecipRate PrecipType essPrecipYesNo
Visibility	essVisibility

3.4 Roadway Surface Condition and Visibility Level Decision Trees

The objective of the decision tree is to document how the selected data elements are used to determine the roadway surface conditions and visibility level at or near the ESS station location. This is done in two steps. A decision tree first considers weather and roadway surface conditions. Once determined, the associate visibility level is estimated using the parameter “essVisibility.” The weather and surface conditions decision tree is shown in Figure 2. The visibility decision tree is shown in Figure 3.

Note in Figure 2 that absorption, error, and unknown surface observations are all "dead end" readings. This is because they do not offer any information that relates to the weather and surface conditions of interest. Error and unknown values contain no information, and "absorption" indicates the presence of

un-dissolved de-icing chemicals on the roadway, which is specific to the location of the sensor. When one of these surface status readings is encountered, the system may be instructed to check precipitation and temperature observations to determine if moisture, ice, or snow is present and proceed accordingly. Obviously, the results obtained without useable surface status information will not be as specific or reliable as those obtained otherwise.

3.5 Changes to Traffic Signal Timing Parameters

With the weather and roadway conditions established as described in section 3.4, traffic signal timing adjustments may be assigned to each condition or group of conditions determined by the decision tree. In the most elementary scenario, as in the case of intersections running on free isolated control mode, an increase in amber/all red clearance interval can be assigned to each incrementally more hazardous set of surface and visibility conditions. More generally, and for coordinated arterial systems, inclement weather timing plans are site specific and should be actuated according to the conditions as determined by the decision tree factoring operational parameters such as speed limit, distances between intersections, and control type. Table 6 lists the possible impact of weather on traffic flow parameters at signalized intersections as reported in the literature [see Perrin et al. 2002, Nakatsuji 2003, Sterzin 2004, Hranac et al. 2006, and Lownes 2006]. Table 7 documents the recommended changes to signalized intersections' control parameters under different weather, roadway surface, and visibility conditions. These values were implemented in the weather-responsive traffic signal control decisions used in this study.

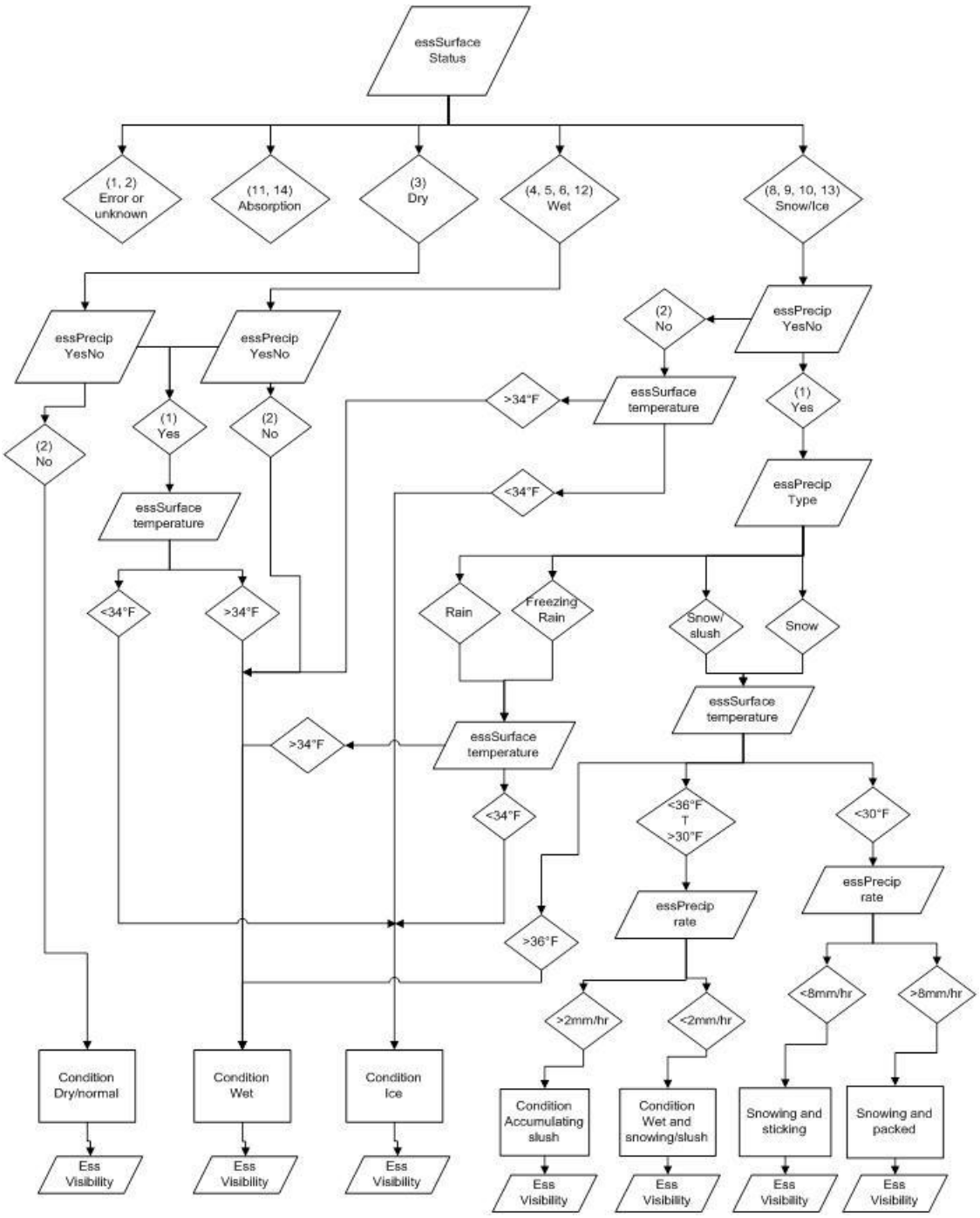


Figure 2. Weather Condition and Roadway Surface Condition Decision Tree

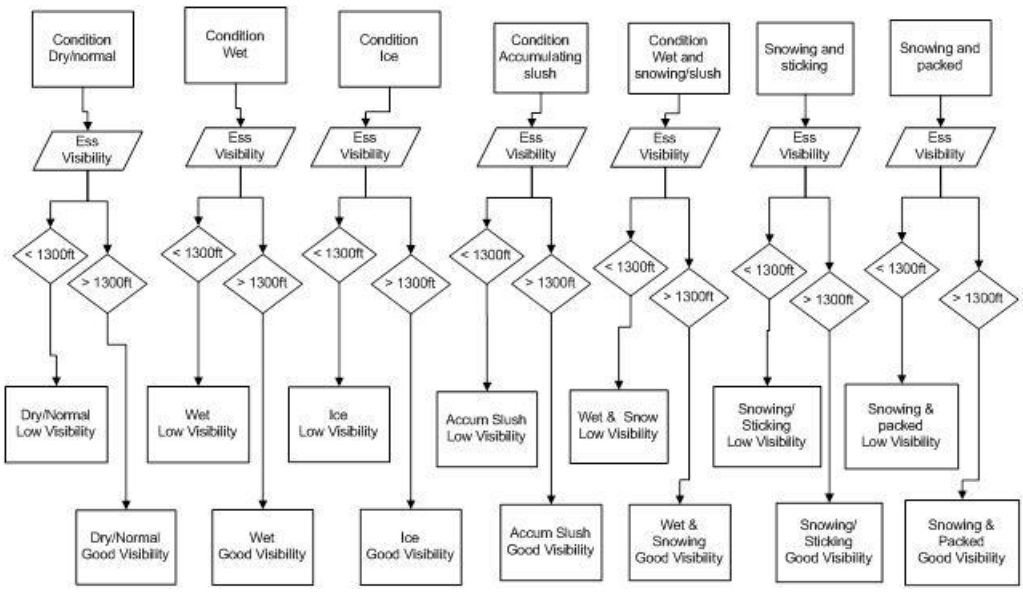


Figure 3. Visibility Level Decision Tree

Table 6. Possible Impact of Weather on Traffic Flow Parameters at Signalized Intersections

Condition	Saturation flow (% Reduction)	Free Flow Speed (% Reduction)	Start-up Lost Time (% Increase)	Deceleration rate
Dry	No change	No change	No change	2.6 m/s^2
Wet	2% - 7%	0% - 8.6%	5%	2.6 m/s^2
Wet and Snowing	7% - 11%	1% - 13%	5%	1.96 m/s^2
Wet and Slushy	15% - 18%	22% - 25%	5%	1.96 m/s^2
Slushy	21% - 20%	28% - 30%	5%	1.96 m/s^2
Snowing and Sticking	11% - 20%	34% - 35%	23% - 50%	1.96 m/s^2
Snowing and Packed	11% - 16%	34% - 42%	23% - 50%	1.96 m/s^2
Temperature < -10 C	1 % - 8%	1% - 2%		
Low visibility (Fog)	10% - 11%	7% - 12%		

Table 7. Recommended Changes to Signalized Intersections' Control Parameters under Different Roadway Surface and Visibility Conditions

Roadway Surface/Visibility condition	Percent increase in amber- and -all-red interval	Changes to coordination control parameters
Dry	No change	Site specific
Wet	10%	Site specific
Wet and snowing	13%	Site specific
Wet and Slushy	22%	Site specific
Slushy in Wheel path	30%	Site specific
Snowing and packed	42%	Site specific
lowest friction (black ice)	50%	Site specific
Low Visibility	10% - 15%	Site specific

Chapter 4. Potential Safety and Operational Benefits of Weather-Responsive Signal Systems

4.1 Overview

The survivable weather-responsive traffic signal system developed as part of this project was evaluated and tested by conducting two analyses: traffic system benefits analysis and software testing and risk analysis. A HILS model was used to assess the operational and safety benefits of adjusting a signal timing plan. Traffic safety benefits were assessed through the use of surrogate measures such as the number and type of conflicts due to weather effects. Software testing and risk analysis provided three critical results: 1) potential risks were identified related to system operation, 2) the consequences of faults were assessed, and 3) risk mitigation strategies were provided through the implementation of "design-for-survivability" software development. The results of the software testing and risk analysis are provided in Chapter 5. This chapter focuses on the potential safety and operational benefits of weather-responsive traffic signal systems.

4.2 Hardware-in-the-Loop Testing Environment

A HILS model was used in the system testing and verification analysis. The HILS model used in this project included 1) a workstation running VISSIM microscopic simulation model, 2) an Econolite ASC/3 traffic controller, 3) a CID to facilitate the exchange of data between the simulation model and the traffic controller, and 4) an external processing unit that runs both the software application and the weather-responsive control algorithm and is also connected to both the Clarus system and the traffic controller. Details of the HILS model and its components are shown in Figure 4 and Figure 5. The microscopic simulation model generates and models the traffic and detector data for the network. The processing unit receives relevant environmental data from the Clarus system, processes it, and communicates control decisions to the traffic controller through the Ethernet port. The traffic controller used in the model was selected to match the controllers used in the city of Moscow, Idaho, traffic signal system; however, the type of controller used did not affect the software or the system functionalities as all communication from the processing unit to the controller followed the NTCIP communication standards. The system did not have any restrictions on use regarding any proprietary data.

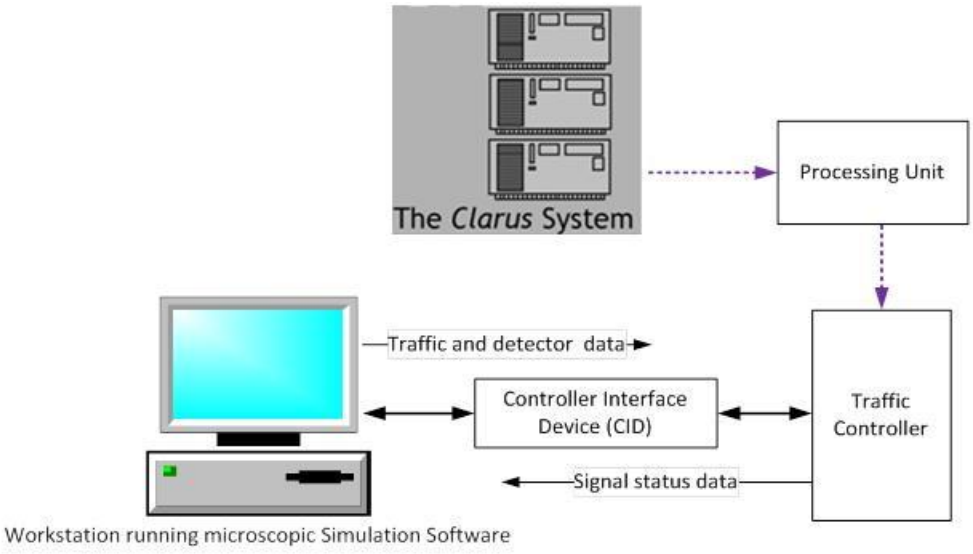


Figure 4. Hardware-in-the-Loop Model Components

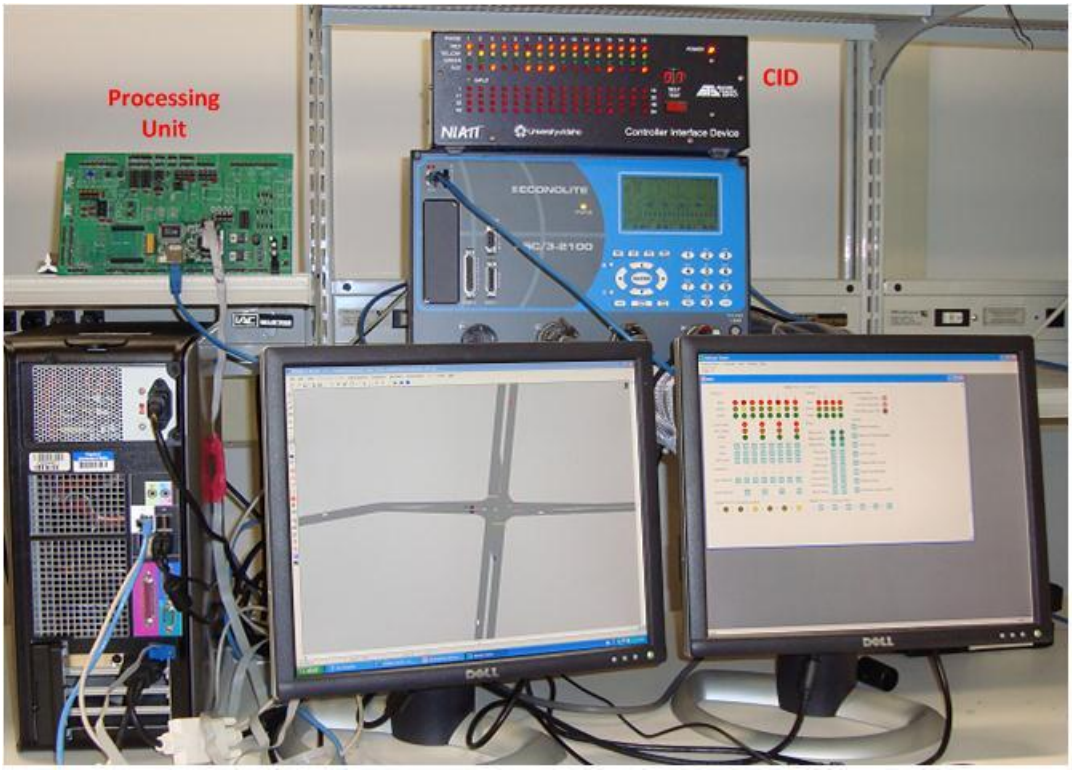


Figure 5. Hardware-in-the-Loop Model Laboratory Setting - University of Idaho's traffic controller laboratory – Moscow, Idaho

[University of Idaho photo, used with permission]

4.3 Weather-Specific Car-Following Parameters

Several VISSIM car-following parameters can be adjusted to describe weather-specific driving conditions. Table 8 shows the Wiedemann 99 car following model parameters and their suggested values [Lownes 2006]. This model is the car-following model used in VISSIM to model car-following behavior along arterial networks.

Table 8: Car Following Parameters (Wiedemann 99 Model)

Parameter	Description	Range of Suggested Values	
		Minimum	Maximum
CC0	Desired distance between stopped vehicles (m)	0.61	3.05
CC1	Headway Time (secs)	0.50	1.50
CC2	Following Distance Oscillation (m)	1.53	6.10
CC3	Entering Following	-15.00	-4.00
CC4	Negative Following Threshold	-2.00	-0.1
CC5	Positive Following Threshold	0.10	2.0
CC6	Speed Oscillation	2.00	20.0
CC7	Acceleration Oscillation (m/sec^2)	0.15	0.458
CC8	Stopped Acceleration (m/sec^2)	1.95	3.05
CC9	Acceleration at 50 mph (m/sec^2)	0.64	2.29

To model driving behavior under different roadway surface and weather conditions, adjustments need to be made to several of these car-following parameters as well as to other vehicle and driver characteristics. These adjustments are discussed in the following bullets:

- CC0 represents the desired distance between stopped vehicles. The default value for this parameter is 1.50 meters, and this value was kept the same for all weather conditions as this parameter is not likely to be impacted by changes in weather or road surface condition.
- CC1 is the desired headway between vehicles. Saturation flow rate was used to determine values for CC1 for different roadway surface and weather conditions. For heavy rain situation, it is estimated that there will be a 10 percent decrease in the saturation flow rate [Hranac et al. 2006]. Assuming a base saturation flow rate of 1,900 vehicles per hour per lane (vphpl) during clear and dry weather conditions, the 10 percent reduction in flow gives a saturation flow rate of 1,710 vphpl during rainy conditions. This corresponds to a saturation headway of 2.11 seconds. This value is slightly higher than the value suggested by Lownes [2006]; however, it is more descriptive of driving in heavy rain. Similarly, saturation flow rate will decrease by 15 percent and 20 percent for snow and ice conditions, respectively. This corresponds to a CC1 value of 2.23 seconds and 2.37 seconds, respectively.
- CC2 is the variation in the safe following distance. Higher CC2 values indicate more cautious drivers while lower values represent more aggressive drivers. CC2 values of 3.05 meters, 4.57 meters, and 6.10 meters were used in VISSIM to describe rain, snow, and ice conditions, respectively.
- CC3 is the parameter that defines the time before a vehicle enters into a following mode. Lownes [2006] concluded that CC3 has little impact on the capacity of roadways since it

does not control the deceleration rate or the acceleration rate of the vehicle. The default value for this parameter is (-8.00), and this value was kept the same for all weather conditions.

- CC4 and CC5 describe the sensitivity of drivers' reaction to changes in the leading vehicle's speed. Low values indicate a high sensitivity to leading vehicles while high values indicate the opposite. For wet roadway surface conditions, values of -0.50 and 0.50 were used for CC4 and CC5, respectively. For snow conditions, values of -0.75 and 0.75 were used for CC4 and CC5, respectively. For icy roadway surface conditions, values of -1.00 and 1.00 were used for CC4 and CC5, respectively.
- CC6 describes the effect of the distance between vehicles on the range of possible speeds (speed oscillation). The impact of this parameter is negligible unless the speed oscillation is great [Lownes 2006]. The default value for this parameter is 11.44 meters, and this value was kept the same for all roadway surface and weather conditions.
- CC7 describes the oscillation of the acceleration of vehicles. Lownes [2006] indicates that this parameter has little impact on traffic flow. The default value for this parameter is 0.25 meters/sec^2. This value was kept the same for all weather conditions.
- CC8 is the acceleration rate of a vehicle beginning from a stopped position. For a wet pavement condition, a CC8 value of 1.95 m/sec^2 was used, representing the lower boundary of the value range suggested in the VISSIM user's manual [PTV 2010]. For snow and ice conditions, a value of 1.77 m/sec^2 was used. This value is based on the 95th percentile acceleration values obtained from Nakatsuji [2003].
- CC9 does not apply in the situation since the model tested was for a network with speeds less than 50 mph.
- Vehicle speeds during rainy conditions can experience a decrease that ranges from 10 percent to 25 pecent [Sterzin 2004]. This study used an average decrease in speeds of 20 percent. For snowy weather conditions, data from a study by Perrin et al. suggests a reduction of 35 percent in desired speed [2002]. For icy road surface conditions, the desired speed will be reduced by 42 percent, which is the upper range of speed reductions as suggested by Perrin et al. [2002]. These speed reduction percentages were applied to the linear speed distribution in VISSIM for speed values that range from 25 mph to 45 mph.
- The desired acceleration and deceleration in snowy and icy conditions were measured in a study that was done by Nakatsuji et al. [2003]. They placed sensors on vehicles to observe the longitudinal and lateral accelerations of vehicles in wintery weather conditions. They generated a distribution of accelerations and decelerations from which average acceleration and deceleration values were derived (Figure 6 and Figure 7, respectively). For this study, desired acceleration and deceleration values of 0.68 m/sec^2 and 0.58 m/sec^2, respectively, were used for both snowy and icy weather conditions. For rainy conditions, the values of 1.97 m/sec^2 and 1.68 m/sec^2, respectively, were used.
- The maximum deceleration rate for vehicles with anti-lock braking systems during rainy weather condition was set at a value of 6.98 m/sec^2 as suggested by Fambro et al. [2000]. For snowy and icy conditions, the maximum deceleration rates used in this study were 2.56 m/sec^2 and 2.07 m/sec^2, respectively. These values are based on the research completed by Lu [1996] where maximum deceleration values were measured for different vehicles using different types of tires. The maximum deceleration rates used in this study were for vehicles with studded tires.

Weather-specific VISSIM car-following model parameters for different roadway surface and weather conditions are listed in Table 9.

Table 9. Weather-Specific VISSIM Model Parameters

Parameter	Weather Condition		
	Rain	Snow	Ice
CC0 (m)	1.50	1.50	1.50
CC1 (seconds)	2.11	2.23	2.37
CC2 (m)	3.05	4.57	6.10
CC3	-8.00	-8.00	-8.00
CC4	-0.50	-0.75	-1.00
CC5	0.50	0.75	1.00
CC6	11.44	11.44	11.44
CC7	0.25	0.25	0.25
CC8 (m/sec^2)	1.95	1.77	1.77
CC9	0.64	0.64	0.64
Reduction in free flow speed (percentage)	20	35	42
Average Desired Acceleration (m/sec^2)	1.97	0.68	0.68
Average Desired Deceleration(m/sec^2)	1.68	0.58	0.58
Maximum Deceleration (m/sec^2)	6.98	2.56	2.07

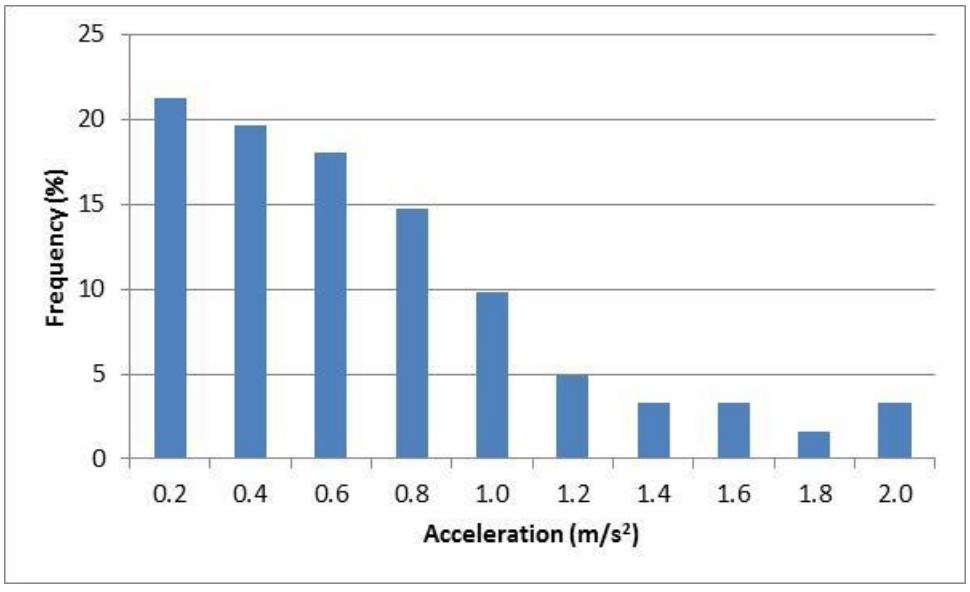


Figure 6. Acceleration Distribution under Snowy and Icy Conditions, [Nakatsuji 2003]

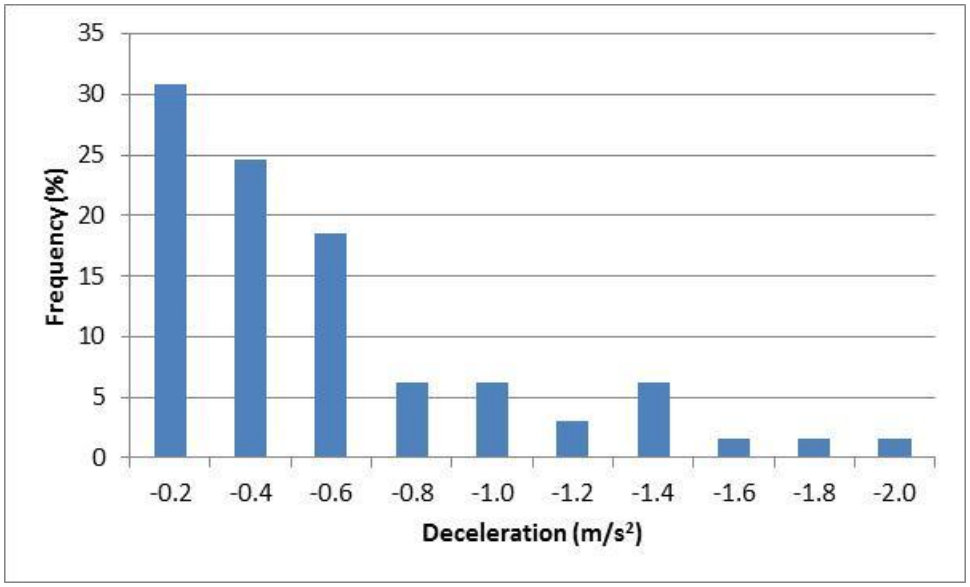


Figure 7. Deceleration Distribution under Snowy and Icy Conditions, [Nakatsuji 2003]

4.4 Surrogate Safety Assessment Model Algorithm

The Federal Highway Administration (FHWA) released the Surrogate Safety Assessment Model (SSAM) to assist in estimating the safety of roadways using microscopic simulation modeling. SSAM uses a trajectory file (TRJ) generated by VISSIM to quantify the number and type of conflicts that occur during the network operations. SSAM compiles data from the TRJ file through the following four steps [Gettman 2008]:

- The first step involves determining the analysis area dimensions, which define the size of the zones SSAM uses. Depending on the units that VISSIM or other simulation software use, SSAM divides the analysis area into smaller zones. These zones can be as small as 15.25 meters by 15.25 meters (50 feet by 50 feet).
- The second step involves analyzing a single time step of the TRJ. The expected locations of each vehicle in the analysis zone are projected as a function of the vehicle's speed up to the user-defined time-to-collision (TTC) value. The path that the vehicle follows is based on the next 10 seconds of trajectory data. The projected travel distance is estimated by first gathering the vehicle's kinematic data, such as location, speed, and acceleration at a single time step and several subsequent time steps. All vehicles are defined as polygons, and then the distance that the vehicle will travel is calculated as:

  Equation 1: $$DIS_1 = V_1 * MaxTTC$$

  Where DIS_1 is the distance traveled during the first time interval, V_1 is the velocity at the time step, and *MaxTTC* is the user-defined TTC (meaning that TTCs above this value are not considered conflicts). The vehicle's location for the next time step is calculated as:

Equation 2: $$DIS_2 = |Location\ (t+1) - Location\ (t)|$$

Where DIS2 is the distance traveled during the second time interval and *Location (t*) and *Location (t+1)* are the vehicle's location at times t and t+1, respectively. If DIS_1 is greater than DIS_2, DIS_2 is subtracted from DIS_1 and Equations 1 and 2 are applied again until DIS_2 is greater than DIS_1. Once DIS_2 is greater, that point is used as the projected location of the vehicle.

- The third step involves projecting the vehicle's rectangular shape to its future projected position. The rectangular vehicle shape is overlaid onto the grid that was defined in the first step. The vehicle is counted as an occupant of each of the square zones within which it resides. SSAM then checks for overlap within each zone of the polygons that define a vehicle. If an overlap exists, that is counted as a conflict.
- The fourth step involves a refinement process of earlier steps. The TTC of a vehicle pair is iteratively shortened so that a more precise actual TTC can be estimated. The lowest TTC value that still produces an overlap of the vehicle polygons is reported as the TTC for the respective vehicle pair. This process allows users to differentiate between conflicts that do not end in a crash and conflicts that will end in a crash. If the vehicle rectangles do not overlap between projection times between 0 and the MaxTTC, then the vehicle pair is analyzed based on post-encroachment time (PET). This time is based on a following vehicle occupying the same space as a leading vehicle. If the following vehicle occupies the same space within the user-defined PET, the vehicle pair is kept in the conflict list until it is evident that the PET will not reduce enough to reach the MaxTTC.
- SSAM outputs several parameters that can be used in the analysis of conflicts. These parameters include x-y coordinates of vehicles, differences in speeds or acceleration, vehicle position, and simulation time. Key parameters that are reported include conflict angle, conflict type, and TTC. The user can define the breaking points between the different conflict types, which include lane changing, rear-end, and crossing conflicts.

FHWA analyzed the practicality of the SSAM software through theoretical validation and field validation. The theoretical validation consisted of utilizing several different software models, including VISSIM, to verify if SSAM could statistically distinguish differences between different intersection types. This validation found that SSAM was able to clearly discern a difference in the number of conflicts, type of conflicts, and severity of conflicts between different intersection designs. The field validation compared intersections modeled with VISSIM with historic crash data. This validation showed correlation except for path-crossing conflicts, which were underrepresented. The R-squared value associated with the model was 0.41, which is consistent with other studies that predicted crashes at similar intersections. The study also found that volume-based prediction models correlated more closely to actual crash data than SSAM's predictions. Even though volume-based prediction models relate better to field data than SSAM, the relative difference in conflict amounts between the different weather conditions tested in this study is statistically significant and emphasizes SSAM's ability to predict conflicts [FHWA 2011].

4.5 Simulation Model Analysis – Base Condition

The simulation model used in this part of the analysis represents a signalized intersection in the city of Moscow, Idaho. The major road in the intersection is US95, a 4-lane highway that runs north-south with a speed limit of 35 mph. The minor road is Palouse River Drive, a two-lane highway also with a speed limit of 35 mph. The geometric characteristics of the intersection are presented in Figure 8.

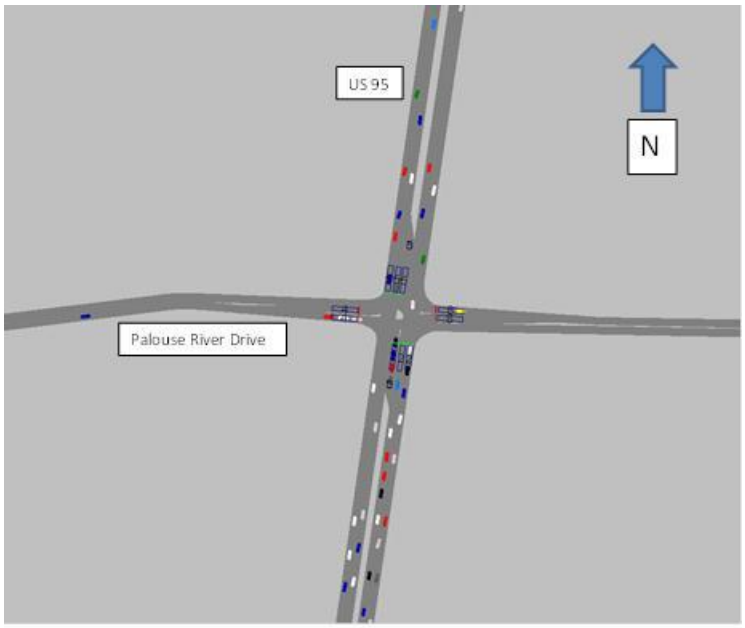


Figure 8. Simulation Model Used in the Analysis

The intersection is controlled by an actuated controller operating in a free (uncoordinated) mode with the following control parameters:

- Yellow and all-red clearance interval (both approaches) = 4.8 seconds
- Minimum green time (both approaches) = 5.0 seconds
- Maximum green time (both approaches) = 40 seconds
- Vehicle extension (both approaches) = 2.0 seconds

Three different major road volume levels were considered in the analysis: low volume (300 vphpl), moderate volume (700 vphpl), and high volume (1,100 vphpl). Volume for the minor road was kept constant at 500 vehicles per hour per lane. Models were developed to represent four different weather scenarios: dry, heavy rain, snow, and ice. For each weather scenario, VISSIM car-following model parameters were adjusted using the weather-specific parameter values listed in Table 9. Each of the twelve volume-level and weather scenario cases was run ten times using different random seed numbers. The duration of each simulation run was 3,900 seconds. No data was collected during the first 300 seconds of the simulation. Surrogate safety measures were obtained from vehicle trajectory files using the FHWA's SSAM tool.

Figure 9 shows average intersection approach delay under different weather conditions for the three volume levels. The average number of stops for the three volume levels are presented in Figure 10. The results show an expected trend. Both measures (average intersection delay and average number of stops) increased significantly during both snow and ice conditions. For low volume condition, the average intersection delay increased from 8.7 seconds/vehicle during dry weather conditions to 22.1 seconds/vehicle during snow conditions and to 27.2 seconds/vehicle during icy roadway surface conditions. A similar significant increase is observed in the number of stops. This pattern is consistent in the three volume levels examined in this study. For high-volume conditions, the average delay and number of stops during icy conditions were slightly lower than those for snowy conditions. This can be attributed to fewer stops during icy conditions at this volume level.

The total number of conflicts and the type of conflicts under different weather conditions are presented in Figure 11 and Figure 12, respectively. The results show that, for high volume, the total number of conflicts increased significantly from 340 conflicts during dry weather conditions to 728 conflicts during heavy rain conditions. This value is 632 conflicts during snowy conditions and 447 for icy conditions. For moderate volume conditions, the number of conflicts during dry and rain conditions was marginal (less than 50 conflicts), jumped to 610 conflicts during snowy conditions and to 361 conflicts during icy conditions. The number of conflicts seems to be very sensitive to volume level, speed and acceleration values, and to the value of TTC used in the conflict analysis. The type of conflicts results, shown in Figure 12, reveal that rear-end conflicts are the most common type that occur during rainy, snowy, and icy weather conditions. The effectiveness of weather-responsive traffic signal systems in improving safety (reducing the number of conflicts) and efficiency (reducing delay and stops) during inclement weather conditions was tested using these base-condition delay, stop, and conflict data. The results of these comparative analyses are presented in the next section.

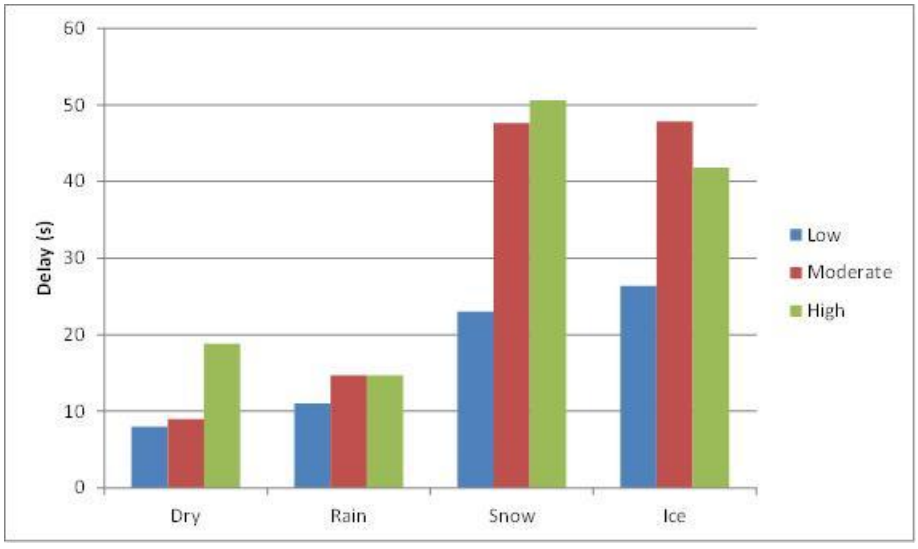


Figure 9. Average Intersection Delay for Different Weather Conditions

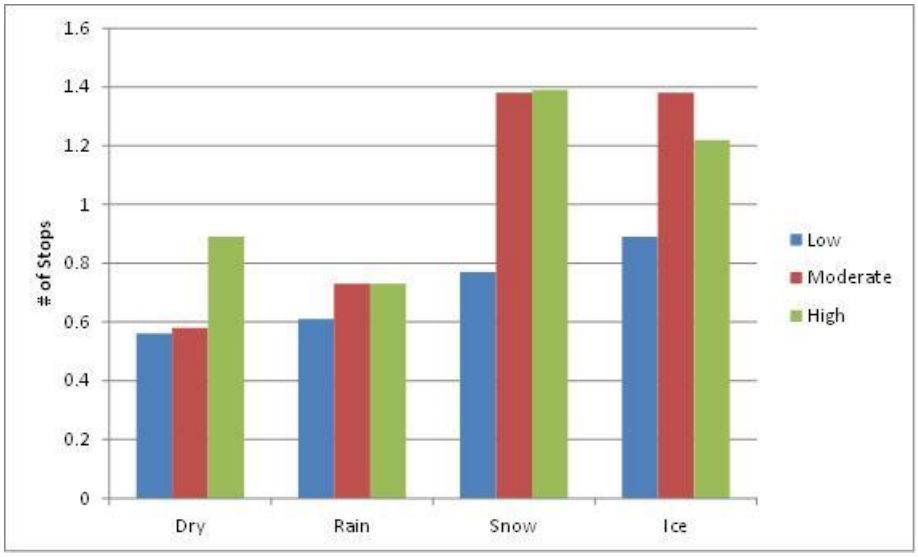


Figure 10. Average Number of Stops for Different Weather Conditions

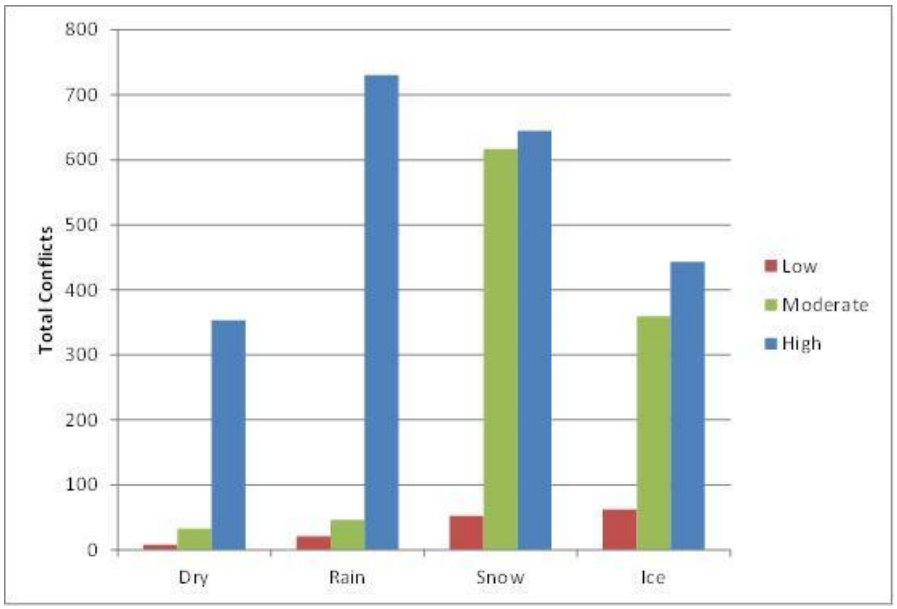


Figure 11. Total Conflicts under Different Weather Conditions

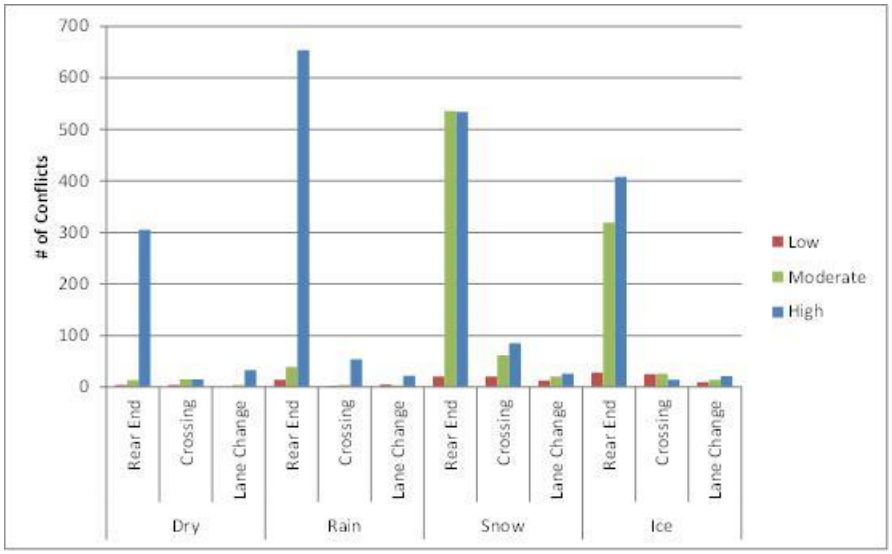


Figure 12. Types of Conflicts under Different Weather Conditions

4.6 Potential Safety and Operational Benefits of Weather-Responsive Signal Systems

In this part of the analysis, simulation models for rain, snow, and ice conditions were run with signal timing parameters adjusted to reflect the changes in the characteristics of traffic operations during these inclement weather conditions. The signal timing parameters used with different weather conditions are presented in Table 10.

Table 10. Signal Timing Parameters for Different Weather Conditions

Signal Timing Parameters	Weather Condition			
	Dry	Rain	Snow	Ice
Yellow and All red (seconds)	4.8	5.3	6.8	7.2
Minimum Green (seconds)	5.0	5.5	7.2	7.5
Vehicle Extension (seconds)	2.0	2.2	2.8	3.0
Maximum green (seconds)	40	40	45	50

The rain, snow, and ice simulation models, with the weather-adjusted signal timing parameters, were run ten times using different random seed numbers. This was done for the three volume levels considered in the analysis. The duration of each simulation run was 3,900 seconds. No data was collected during the first 300 seconds of the simulation. Again, surrogate safety measures were obtained from vehicle trajectory files using the FHWA's SSAM tool.

To assess the potential safety and operational benefits of weather-responsive signal timing parameters, the output of these runs were compared against delay, number of stops, total number of conflicts, and number of rear-end and crossing conflicts obtained from the base conditions runs, with no signal timing adjustments. The results of these comparisons are presented in Table 11. The percent change in each of these measures represents the difference between the base condition and weather-adjusted values divided by the base condition value and multiplied by 100. Positive values indicate reduction (improvements) as a result of the weather-responsive signal timing plan implementation.

Table 11. Percent Reduction in Delay, Stops, and Conflicts as a Result of Weather Adjusted Signal Plan

	Low Volume			Moderate Volume			High Volume		
	Rain	**Snow**	**Ice**	**Rain**	**Snow**	**Ice**	**Rain**	**Snow**	**Ice**
Average delay	2.31	4.88	3.49	6.32	9.59	7.42	8.63	12.64	11.09
Number of stops	4.82	7.32	5.19	7.11	8.74	8.69	9.60	14.32	12.63
Total conflicts	---*	6.14	5.81	11.75	18.13	20.44	14.84	33.19	39.78
Rear-end conflicts	---*	---*	---*	9.41	21.33	23.18	11.39	42.12	43.68
Crossing conflicts	---*	---*	---*	5.94	9.18	13.26	8.14	20.72	18.94

*marginal number of conflicts in the base condition

The results presented in Table 11 reiterate the potential safety and operational benefits of weather-responsive traffic signal systems. The potential crash reduction benefits, expressed as the percent reduction in total, rear-end, and crossing conflicts, seem to be higher during snow and ice weather conditions. The potential crash reduction benefits increases as the volume level increases. Rear-end conflicts are the conflict type most eliminated by a weather-responsive traffic signal system with a potential average reduction of approximately 22 percent for moderate volume levels and 43 percent for high volume levels. The weather-responsive signal timing plans also showed a considerable reduction of both delays and stops. Again, the percent reduction increases as the volume level increases. While these results are based on microscopic simulation modeling and surrogate safety measures, they still provide a reasonable assessment of the crash reduction potential of weather-responsive traffic signal systems. The potential reduction in delays and stops resulted from this analysis seem consistent with what has been reported in the literature.

Chapter 5. Software Design and Testing

5.1 Introduction

As the components controlling our critical infrastructures are increasingly relying on networked computing systems this connectivity also becomes the focal point for security and survivability considerations. It is thus more important than ever to include security and survivability starting at the specification and design stage, rather than in an add-on fashion. Design for survivability incorporates this philosophy and will be demonstrated for a typical embedded control application connected to the Internet. This type of system is found in most devices controlling our critical infrastructures.

The software architecture employs two revolutionary new approaches: 1) design for survivability and 2) a measurement-based methodology for embedded systems. Whereas the concepts have been discussed in the fault-tolerance and security community for almost a decade, implementations are limited to academic prototypes, none of which were in traffic signal systems. The main reason is that most systems that would benefit from these approaches already exist and it is uneconomical to retrofit to accommodate these two principles, i.e., the principles are based on integration and not retrofitting.

The project described here could serve as a major milestone in the development of safe and secure transportation systems. First, it is sufficiently small in scope to utilize both approaches in a manageable way. Second, the application is part of a critical infrastructure, therefore justifying the additional complexity and effort. This is very important: most applications that have considered high levels of fault-tolerance have been in the area of ultra-reliable systems, which typically include systems like primary flight control or military applications. However, even these applications are only now realizing the need for survivability in addition to fault-tolerance.

The architecture of the proposed work is a fundamental building block in a highly networked and interactive communications system. As such, it will be exposed to all faults that may occur locally or via the network, ranging from benign component failures to malicious cyber threats. Due to the fact that this is a safety critical system, the design process associated with ultra-reliable real-time systems design must be used. As a result, we propose using the design philosophy called “Design for Survivability” [Krings 2008] to incorporate fault tolerance in a more general way as it not only considers components or software faults, but also faults associated with malicious acts, i.e., maliciously induced faults. In this way, the project is based on a measurement-based methodology for survivability of transportation control system components.

Design for survivability is an approach that has much in common with *Design for Testability*. As integrated circuits became larger, exhaustive testing became infeasible, i.e., the number of test scenarios needed to test circuits became intractable. As a result, it was realized that circuits had to be designed for testability. As systems became increasingly complex and difficult to analyze, the notion of designing for survivability, i.e., integrating the mechanisms that aid survivability into the system (rather

than as an add-on feature), became a natural extension, analogous to design for testability [Krings 2008]. As a result, to achieve this level of survivability, the proposed system software design employed testing in the form of system measurements and self-diagnostics.

5.2 Fault and Threat Space

Our system needed the basic capability of using and generating data, i.e., data imported from Clarus as well as the potential to serve as a sensor (i.e., data provider) for Clarus. To accommodate this need, the system required a secure interface, capable of dealing with basic fault types. There are too many fault sources to list individually and exhaustively. Therefore the notion of fault models is used, capturing the behavior of a fault, i.e., a fault can produce an error that then can lead to a failure. The diversity of faults and their consequences on a system have been the primary motivator for the definition of fault models. A fault model addresses the behavior of the faults and specifies the redundancy levels required to tolerate a single fault type or a mix of fault types. Many different fault models have been proposed over the years ranging from the simple models that make no assumptions about the fault behavior [Lamport 1982], to hybrid fault models considering multiple fault behaviors. The latter considers a mix of faults ranging from benign, symmetric, to asymmetric faults [Thambidurai 1988], with potential transmissive and omissive behaviors [Azadmanesh 2000].

The fault model of Azadmanesh [2000] constitutes the basis for the faults addressed in the proposed system and is the reference in the communications with Clarus. Omission faults were emphasized, because communication with Clarus may be interrupted. Furthermore, value faults (symmetric and asymmetric) such as infeasible or incorrect input or output data were also deemed important, since any of such faults have the potential to decrease safety. One of the main benefits of Clarus is that it considers quality checking as part of the mission [Limber 2011].

Rather than specifying each of the functionalities of the software, we want to focus on the software architecture as it addresses design for survivability and the measurement-based methodology. There are several key technologies incorporated in these two approaches, including functional software specification, measurement-based certification of normal and non-nominal operation, adaptability, diagnosability, real-time predictability, and fail-safe behavior. In short: all the ingredients to run, observe, analyze, and reconfigure a system.

The most important aspect of the software architecture is the derivation/adaptation of the measurement-based approach introduced in Krings et al. [2001] and refined in Munson, Krings, and Hiromoto [2006] to ensure properties of reliability, security and survivability. Intuitively, the application is defined as a basic set of operations, which are expressed by a collection of functionalities. These functionalities are implemented with software modules, e.g., C functions, and instrumented (via instrumentation telemetry) in a way that allows measuring the behavior of the operations, functionalities and modules in real-time. This becomes extremely useful when studying the behavior of individual functionalities during execution. As has been demonstrated in previous work with the behavior of networked systems under attack, normal executions of functionalities can be captured like a fingerprint of that functionality (called functional profiles) [Krings 2001]. Any deviation from such a profile can then be interpreted as an unusual, non-nominal execution. This in turn allows for responsive measures (e.g., changing the execution state, re-executing a functionality or system reconfiguration) as defined by a contingency management system. The basic operation of this approach is shown in Figure 13. The executing program is observed via the instrumentation telemetry. The feedback-loop of 1) observing, 2) analyzing, 3) changing parameters, and 4) controlling the

software design or operation is critical during software design and later during its operation. In the latter case, it allows to implement the survivability measures upon detection of deviations from certified operation, e.g., unusual or undesired operation.

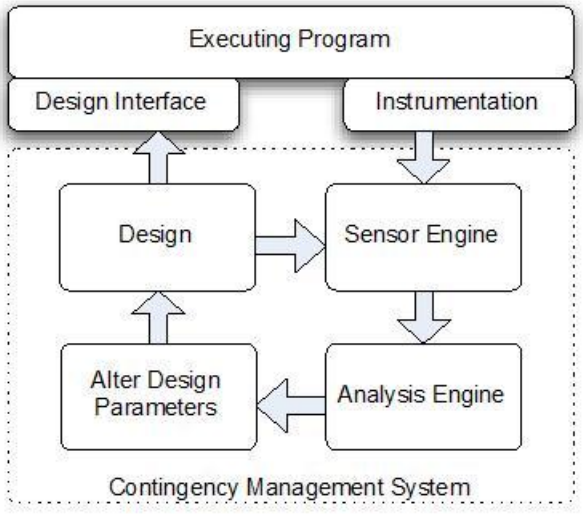


Figure 13. Overview of Measurement-based Design Methodology

The overall system architecture is comprised of multiple components, the executing program, and the contingency management system. The sole purpose of the latter is to watch the execution in real-time and react to unwanted changes as they would occur as the result of system components malfunctioning or unwanted manipulations of the system by intruders and/or hackers.

Should multiple systems be deployed in proximity, then the principle of spatial redundancy could be used to tolerate failures and malicious attempts to manipulate the system. For this situation, agreement algorithms can be used to eliminate the impact of incorrect values and data. For instance, if the system needs to agree on values that represent Clarus data, then exact agreement algorithms can be used, e.g., the early stopping agreement described by Krings and Feyer [1999]. If there are real-valued control parameters that have to be agreed upon approximate agreements can be used [Azadmanesh 2000, 2003].

5.3 Software Architecture

The system connects to the Local Clarus Server (a local mirror site of specific subscription data) or Clarus using the network interface to the Internet. In regular intervals, e.g., local sensor data is typically updated every 5 to 15 minutes, the Clarus data is read and converted by the Rabbit, the desired sensor data is extracted, and specific algorithms are used to compute the yellow timing from the critical extracted parameters. The traffic controller is then updated. All of this is monitored by the Operation Monitoring and Contingency Management System.

Figure 14 shows an overview of the software architecture and its interface to Clarus. Shaded blocks indicate the hardware interfaces. The *Network Interface* represents the connection to the Internet. Because the signal control system has its own data representation, the Clarus data has to be

translated in the *Clarus Data Conversion Interface*. However, the kind of Clarus data required depends on the enabled control algorithm. These algorithms are modular units in the *Algorithm Engine*. The values computed by the algorithms are then used to update the traffic controller. Survivability measures during the design and operation of the system are centered around the *Operation Monitoring and Contingency Management System*, which interfaces to the software system via the instrumentation telemetry. The adaptability and recovery from any unintended or maliciously induced operations/profiles is determined by the survivability policy and is handled by the *Contingency Management System.* Whereas the block diagram of Figure 14 suggests a high level of complexity, the goal of the project is to operate in a low-complexity environment. The relatively small size of the system makes it a perfect candidate to apply the survivability and the measurement-based methodology effectively.

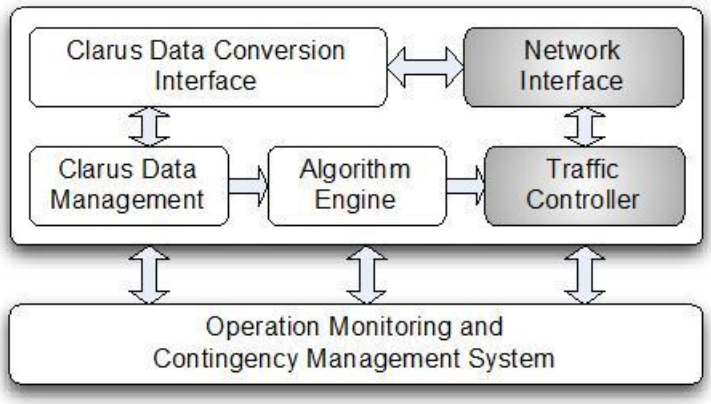


Figure 14. Software Architecture Overview

5.4 Formal Execution Model

During operation of the system, and with proper instrumentation of the software, one can get a "life" picture of how the system is performing in real time, e.g., what the execution of a typical operation looks like, how often functionalities are called by a specific operation, what mix of functionalities is instantiated over a certain window of observation, or how often certain modules get called during a time interval. All of this information is captured in *profiles*. Calling behavior, e.g., operational sequences, is embedded in dependencies identified in static or dynamic precedence graphs, e.g., the call graph of modules.

5.4.1 Principles and Definitions

The notation and general execution model described below are partially adapted and restated from Munson, Krings, and Hiromoto [2006] to suit the more deterministic execution environment of this application. The Rabbit executes a set of operations *O,* with cardinality |*O*|. These operations constitute the *operational machine*. The transition from one operation to another marks an *operational epoch*. Each operation o_i uses one or more functionalities f_j from a set *F* of functionalities with cardinality |*F*|. Similar to the operational epoch the functional epoch is defined by transitions from one functionality to another. Functionalities are implemented by code modules written in Dynamic C, which is a C-like language with a unique multitasking environment (as will be described later). The set of

modules M of cardinality $|M|$ is thus the implementation of the functionalities in code. The frequency spectrum of operations, functionalities, and modules define the *operational, functional* and *module profile* respectively. These profiles will be used later to define certified operations.

The relationship between operations, functions, and modules is defined by a graph G^{OFM}, where the superscript simply indicates that the graph maps from O to F to M. An example is depicted in Figure 15, which shows three operations o_1, o_2 and o_3. The operations utilize specific functionalities, e.g., o_1 uses functionalities f_1 and f_2. Incidentally, f_2 is also used by o_3. The functionalities are implemented by Dynamic C modules, e.g., f_3 is implemented by module m_4, whereas f_4 is realized by m_4, m_5, and m_6.

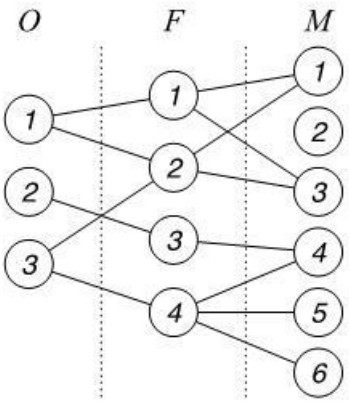


Figure 15. OFM Mapping: Mappings in ($O \times F \times M$)

5.4.2 Profiles

Staying consistent with the notation of Munson, Krings, and Hiromoto [2006] we used letters u, q and p for operational, functional and module profiles respectively. Let u_l denote the probability that the system is executing operation o_l. Then $\boldsymbol{u} = \langle u_1, u_2, \ldots, u_{|O|} \rangle$ is the operational profile of the system.

During execution of the system we are interested in observing the operational profile over n epochs. This observed profile is $\hat{\boldsymbol{u}} = \langle \hat{u}_1, \hat{u}_2, \ldots, \hat{u}_{|O|} \rangle$, where $\hat{u}_i = c_i / n$ is the fraction of system activity due to operation o_i, and c_i is the count of invocations of o_i. As the system activity is continuously monitored, which implies that operational profiles are generated and analyzed, we want to keep track of these profiles. Let $\hat{\boldsymbol{u}}^{\mathbf{k}}$ denote the k^{th} operational profile. Thus $\hat{\boldsymbol{u}}^{\mathbf{k}}$ is observed over n operational epochs, which was preceded by $\hat{\boldsymbol{u}}^{\mathbf{k-1}}$, observed over the previous n operational epochs, and so forth.

Just as in Munson, Krings, and Hiromoto [2006], if we consider m sequences of n epochs each, we can define a centroid $\bar{\boldsymbol{u}} = \langle \bar{u}_1, \bar{u}_2, \ldots, \bar{u}_{|O|} \rangle$, where

$$\bar{u}_i = \frac{1}{m} \sum_{j=1}^{m} \hat{u}_i^j$$

and the distance from $\hat{\boldsymbol{u}}^{\mathbf{k}}$ from centroid $\bar{\boldsymbol{u}}$ is given by

$$d_k = \sum_{i=1}^{n} (\overline{u}_i - \hat{u}_i^k)^2$$

Observed profiles, and how they deviate from the centroid, will be analyzed in subsection 5.4.5.

5.4.3 Dependencies

Whereas the example in Figure 15 shows the relationship between operations, functionalities, and modules, it does not contain any information about dependencies of operations in *O*, functionalities in *F*, or modules in *M*.

The relationship between operations is defined by graph $G^O = (O, \angle)$, *where* $\angle$ (in our application) defines a partial order relation on the operations in *O*, i.e., if o_j depends on o_i then $(o_i, o_j) \in \angle$. In the example of Figure 15, if o_1 is the operation "obtain data," o_2 is "analyze data," and o_3 is "adjust controller," then the logical dependencies among the operations are $o_1 \angle o_2$ and $o_2 \angle o_3$. Any violation of the partial order indicates a problem in the control flow of the program.

We define similar graphs for functionalities and modules; however, the precedence relation, denoted by <, in those cases is a general precedence relation and not necessarily a partial order, e.g., the graph may not be acyclic. Thus $G^F = (F, <)$ and $G^M = (M, <)$ are the graphs defining calling relationships between functionalities and modules respectively. It should be noted that G^M is the static call graph of modules in *M.* Furthermore, the difference in precedence relations should be noted, i.e., $\angle$ denotes a partial order relation, whereas < in general does not. The operational, functional, and module dependency graphs are used to detect invalid transitions.

5.4.4 Dispatching Model

The Rabbit system uses a single processor in which multitasking is implemented using a model defined by *costatements*. A costatement is defined as a task in a nonpreemptive multitasking model. The system executes one costatement at a time. Costatements are typically listed in an infinite control loop in the main program. Each costatement has a statement counter, i.e., a program counter, which indicates which instruction of the costatement will execute when it gets a chance to run. Execution is switched from one costatement (of the infinite loop) to the next in a round-robin fashion when the currently executing costatement "yields" to the next costatement using explicit commands, such as *yield*, *abort* or *waitfor*(*event*). Due to these yielding mechanisms the model is based on good behavior. The state of a costatement is called a *costate*. In the discussions to follow, the terms costatement and costate will be used interchangeably.

A model with such task-switching properties executes deterministically, i.e., a task switch is explicitly demanded by the currently executing task: the active costatement. On the other hand this means, however, that it is possible for a costatement to cause starvation by not yielding. To resolve such a situation, mechanisms like watchdogs and timer interrupts can be used. In this case the system deviates from its otherwise nonpreemptive execution model.

As operations, functionalities, and modules are called from within exactly one costatement at a time, it is possible to exactly determine the functionality and module that are being executed on behalf of a specific operation. Thus, the dispatching model results in executions with a high degree of determinism, which is very desirable when working with profiles. The alternative would be profiles that

mix the frequency spectrum from *all* executions together into one inseparable profile. Here, however, we can separate the profiles, or, even simpler, we let each costate have its own profile.

5.4.5 Costate Profiling

The concepts and notations derived in subsection 5.4.2, i.e., the observed profile $\hat{\boldsymbol{u}} = < \hat{u}_1, \hat{u}_2, ..., \hat{u}_{|O|} >$, the k^{th} operational profile $\hat{\boldsymbol{u}}^{\mathbf{k}}$, the centroid $\bar{\boldsymbol{u}} = < \bar{u}_1, \bar{u}_2, ..., \bar{u}_{|O|} >$, and the distance of $\hat{\boldsymbol{u}}^{\mathbf{k}}$ from centroid $\bar{\boldsymbol{u}}$, can now be defined on a costate-basis. This leads to notation $\hat{\boldsymbol{u}}[\alpha]$, $\hat{\boldsymbol{u}}^{\mathbf{k}}[\alpha]$,$\bar{\boldsymbol{u}}[\alpha]$ and $d_k[\alpha]$ respectively, where α indicates the costate, or costate ID number. Thus each costate α has its own profiling, which is not affected by any non-determinism due to costate (task) switching, i.e., profiles of costates do not interfere.

5.4.5.1 Current State of the System:

The current state of the system is defined by a triplet in the cross product ($O \times F \times M$), which indicates what operation, functionality and module is executing. To keep track of the current state of the system, a table S is maintained that, for each costate α, indicates the currently executing o_i, f_j and m_k. Thus each row α of the table indicates the state of costate α, i.e., $S[\alpha] = [o_i, f_j, m_k]$ indicates that in costate α operation o_i is utilizing f_j by executing module m_k. Since the system can only be in one costate at a time, we can tell the exact state of the system by looking at the table entry of the currently executing costate α. This means that by using S one can deterministically map modules to functionalities and functionalities to operations. This makes our profiling more deterministic and thus much less convoluted than in Krings et al. [2001] and Munson, Krings, and Hiromoto [2006].

5.4.5.2 Determination of Active Costate:

Each costate receives a unique costate ID, denoted by α. The state of the system depends on the costate α which is executing. To determine which costate is executing a global variable called *ActiveCostateID* is defined that is set by each costate

1. when the costate starts execution,
2. after a *yield*,
3. after an *abort,*
4. after a *waitfor* statement.

These four options cover each possible way that the costate starts or resumes execution.

5.4.5.3 Updating the Current State of the System:

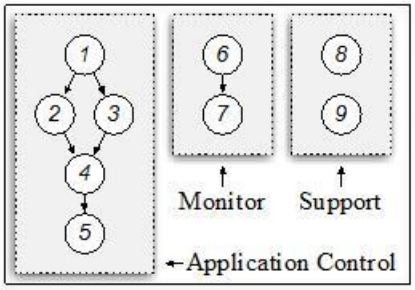
Now that the active costate is known, a module, functionality, or operation knows exactly which costate it belongs to by simply looking at *ActiveCostateID*. For example, if a module m_h is called it can find out which module it was called from by simply looking at $S[ActiveCostateID] = [o_i, f_j, m_k]$, to find out that it was called by m_k as part of functionality f_j, which is used by operation o_i. This knowledge can be used, for example, to check if this module call is consistent with the static call graph G^M, before updating the state table from m_k to m_h, i.e., with m_h now executing we have $S[ActiveCostateID] = [o_i, f_j, m_h]$. Note that a call graph inconsistency would indicate that the program has been altered.

5.4.5.4 Counting Invocations:

The observed profiles result from counting invocations of executions. As shown before, the individual $\hat{u}_i$ of the observed profiles $\hat{\boldsymbol{u}}$ are computed from $\hat{u}_i = c_i / n$. Invocations of operations, functionalities, and modules are counted in c^o, c^f and c^m respectively. Since it is beneficial to separate counts for each costate α, each costate has its own counters $c^o[\alpha]$ of length $|O|$, where $c^o{}_i[\alpha]$ is the count of the i^{th} operation in costate α. Similarly, we define the functionality counters $c^f[\alpha]$ of length $|F|$ with elements $c^f{}_i[\alpha]$ and the module counters $c^m[\alpha]$ of length $|M|$ with elements $c^m{}_i[\alpha]$.

5.4.5.5 Application Overview:

The number of costates in an embedded control system is typically relatively small. Our application software system consists of three significant costates as shown in Figure 16. A forth costate containing system configuration setup is not shown. The individual operations in costates are described to the right of the figure.



1 Get Clarus data
2 Receive data from LCS
3 Receive data from Clarus
4 Analyze Clarus data
5 Adjust controller
6 Monitor analysis
7 Monitor adaptive reconfiguration
8 Time synchronization
9 Support routines

Figure 16. Costates and Operations

The first costate implements the application control, which consists of the software that gets the data, analyzes it, and makes appropriate adjustments to the controller if necessary. The second costate is the monitor. It analyzes the profiling data collected by the instrumentation and, if necessary, will initiate adaptive reconfiguration. The third costate contains independent support operations, e.g., o_8 synchronizes the timer of the system with a NIST time source.

5.4.6 Certified Executions

Certifying behavior per costate is now possible and will be described using module profiles, $\hat{\boldsymbol{p}}^k[\alpha]$, rather than operation profiles $\hat{\boldsymbol{u}}^k[\alpha]$. While p is used for modules, u is used for operations. The distance of the observed costate profiles $\hat{\boldsymbol{p}}^k[\alpha]$ from $\bar{\boldsymbol{p}}[\alpha]$ can be used so that departure beyond it indicates non-certified behavior of costate α. Specifically, a threshold vector $\varepsilon[\alpha] = <\varepsilon_1[\alpha],...,\varepsilon_{|M|}[\alpha]>$ is pre-assigned, which is greater or equal to $\bar{\boldsymbol{p}}[\alpha]$. We define that a profile is nominal if $\varepsilon_i[\alpha] - \hat{p}_i{}^k[\alpha] \geq 0$. Any execution not satisfying the inequality is off-nominal. Alternatively, one can set a threshold scalar ε and use the distance $d_k[\alpha]$ as a measure to detect off-nominal module behavior.

It should be noted that certification in this context deals with the behavior of the program and should not be confused with guarantees for validity of the adjustment values as computed from the analysis of the Clarus data. An incorrect adjustment value is a simple value fault, i.e., a symmetric fault in the fault model. The impact of such fault is limited. An NTCIP-compliant traffic controller does not accept

changes for a specific parameter that are outside of the specified range of that parameter defined in the NEMA TS2 standards [AASHTO 2005]. The upper value of this range could still allow for a denial of service scenario should the incorrect adjustment value be unreasonably high. For example, the acceptable value for the yellow interval as defined by NEMA TS2 standards ranges from a minimum of 3 seconds to a maximum of 25.5 seconds. No values lower than 3 seconds can be input to the controllers eliminating any critical safety risks in the operation. However, setting the yellow interval near the upper end of the acceptable range (i.e. 24 seconds) will result in a significant deterioration in the system operation and may lead to a denial of service fault. Such a case can, however, be dealt with by testing the adjustment value to be in a predefined adjustment range before updating the controller, i.e., a separate range check is performed in addition to the NTCIP-defined range. Value faults could be the result of a program error, e.g., incorrect computation or incorrect Clarus data. The first is addressed by the Operation Monitoring and Contingency Management System. The second is addressed by the Clarus quality checking algorithm [Limber 2010].

5.5 Run-time Monitoring

5.5.1 Instrumentation

There are three types of instrumentation: operations, functionalities and modules. For each the specific steps are described below. However, it should be noted that one can have a mix of instrumentations. For example, if a module also indicates the start of a functionality, then this instrumentation has to be included as well. Thus, in the most complicated case we could have to instrument the beginning of an operation, then the beginning of a functionality, and then a module. Furthermore, the instrumentation has to be in that order.

5.5.1.1 Operation Instrumentation:

When entering an operation o_i in costate α = *ActiveCostateID* the following tasks are performed:

1. Check for violation of partial order relation in G^O.
2. Update $S[\alpha]$ to indicate o_i is now the current operation, i.e., $S[\alpha]=[o\_i,-,-]$, where - indicates no change.
3. Increment the frequency count $c^o_i[\alpha]$\$ to account for the instantiation of o_i.

5.5.1.2 Functionality Instrumentation:

When entering a functionality f_i in costate α = *ActiveCostateID* the following tasks are performed:

1. Check for violation of partial order relation in G^F.
2. Check for violation of mappings in G^{OFM}, i.e., determine if the execution of f_i is consistent with the operations in the graph.
3. Update $S[\alpha]$ to indicate f_i is now the current functionality, i.e., $S[\alpha] = [-,f_i,-]$.
4. Increment the frequency count $c^f_i[\alpha]$ to account for the instantiation of f_i.

5.5.1.3 Module Instrumentation:

When entering a module m_i in costate α = *ActiveCostateID* the following tasks are performed:

1. Check for violation of precedence relation in G^M.
2. Check for violation of mappings in G^{OFM}, i.e., determine if the execution of m_i is consistent with the operation and functionality in the graph.
3. Update $S[\alpha]$ to indicate m_i is now the current module, i.e., $S[\alpha] = [-,-,m_i]$.
4. Increment the frequency count $c^m{}_i[\alpha]$ to account for the instantiation of m_i.

5.5.2 Experimental Results

A prototype was built based on a Rabbit MiniCore RCM5700, which incorporates the Rabbit 5000 microprocessor with integrated 10/100Base-T Ethernet funcitonality and 128KB of onchip SRAM. The Rabbit runs Dynamic C version 10.5.4, which has been instrumented to allow operation, function and module profiling. Furthermore, at each level of abstraction the precedence constraints can be validated. The current system utilizes 395 modules, of which 177 are written in Dynamic C and 218 in assembler code. All Dynamic C modules were instrumented. A partial sample profile of the system is given in Figure 17 in which 46 significant modules are represented in four costates.

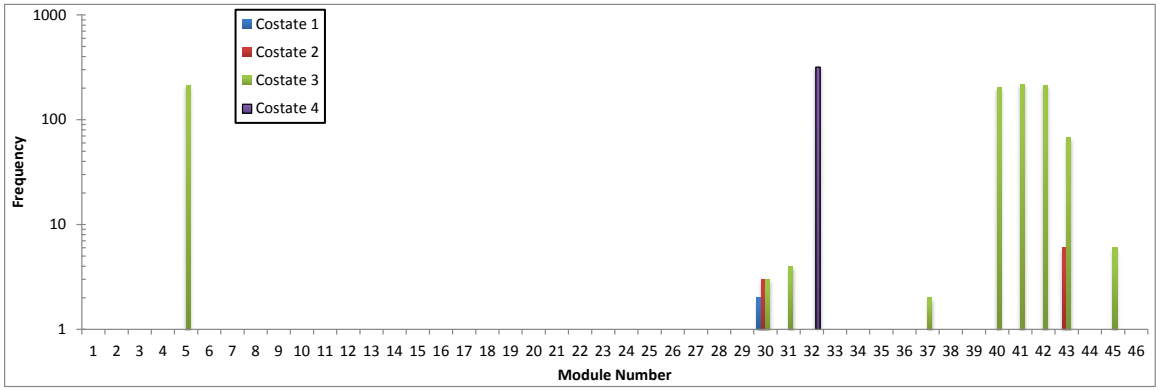


Figure 17. Sample Frequency Counts

Current efforts focused on generating profiles and evaluating them in real-time in order to determine reasonable certification thresholds. Whereas instrumentation for operation and functionalities was implemented, actual testing and evaluating has focused mainly on modules. This strategy was used in order to optimize the learning curve, given the realities of the project duration. Furthermore, some features of dependency modeling have turned out to be more challenging than originally foreseen. Specifically, instrumenting Dynamic C library modules has been limiting in that assembler modules could not be instrumented. This, however, had implications when such modules call Dynamic C modules, i.e., call graph dependencies, when assembler modules call Dynamic C modules, cannot be validated. An approach was taken to deal with this problem by treating validation of violations as a sensor input, similar to the frequency counters. Thus, dependency violations thresholds have to be evaluated.

5.6 Software Design Conclusions

Real-time monitoring of executions of the operational and functional machines, as well as modular profiling, have been explored in order to aid in 1) the design of embedded systems and 2) in the reconfiguration upon detecting deviation from certified behavior. The formal model was introduced and expanded to take advantage of the decrease in non-determinism of executions in the costate task management paradigm.

Chapter 6. Conclusion

State-of-the art software engineering was employed to generate a survivable, reliable, and secure prototype weather-responsive traffic control system for a signalized intersection. The system utilizes a network connection with the Clarus database to obtain weather information and a connection with the traffic controller to obtain and update signal timing plan parameters. Reliability, security, and survivability are achieved by 1) defining normal operations in terms of profiles of a measurable statistic, 2) adopting a task dispatching model that specifies deterministic task execution, 3) utilizing software instrumentation of the tasks to provide real-time profiles, and 4) selecting and executing contingency plans. Minimal hardware requirements exist for the prototype; the off-the-shelf microprocessor, access to power, and a connecting cable are entirely sufficient. The prototype design is such that it would function for any field traffic control application where the overall process can be distilled to predictable tasks. Current traffic control technology supports the proposed system development. Microprocessor traffic controller NTCIP-based communications were tested verifying that the necessary read and write capabilities are available from the microprocessor to any NTCIP-compliant traffic controller.

Development of the prototype followed a standard systems engineering process that included six steps: reviewing the resources, defining the system specifications, designing the system, creating the data interface and analyzing the data, developing the testing environment, and performing verification and timing analysis. The weather data is accessed through a subscription to the Clarus system web interface. Different observation types reported in the Clarus data system are used to determine air and surface temperature, roadway surface condition status, precipitation type and rate, and visibility level at or near the environmental sensing station. The availability and accuracy level of the weather data reported in the Clarus system provided reliable estimates of the weather, road surface condition, and visibility level.

The survivable weather-responsive traffic signal system developed as part of this project was evaluated and tested by conducting two analyses: traffic system benefits analysis and software testing and risk analysis. The potential crash reduction benefits, expressed as the percent reduction in total, rear-end, and crossing conflicts, are highest during snowy and icy weather conditions. The potential crash reduction benefits increase as the traffic volume level increases. Rear-end conflicts are the conflict type projected to be most eliminated by a weather-responsive traffic signal system with a potential average reduction of approximately 22 percent for moderate volume levels and 43 percent for high volume levels. The weather-responsive signal timing plans also show considerable potential in reducing traffic delays and stops. Again, the percent reduction increases as the traffic volume level increases. The potential reduction in delays and stops seems consistent with what has been reported in the literature.

The overall system architecture is comprised of multiple components, the executing program, and the contingency management system. The sole purpose of the latter was to watch the execution in real-time and react to unwanted changes as they would occur as the result of system components malfunctioning or communication failure. Survivability measures during the design and operation of the system were centered around the Operation Monitoring and Contingency Management System, which interfaces to the software system via the instrumentation telemetry. The adaptability and

recovery from any unintended or maliciously induced operations/profiles was determined by the survivability policy and was handled by the Contingency Management System. Because the proposed system has very similar computational requirements to other field traffic control applications, it serves as a major milestone in the development of secure and dependable real-time traffic control systems.

Detecting a system's departure from nominal behavior due to faults or malicious acts has been a challenge to the security and survivability research community for years, and most researchers believe too little progress has been made to counter malicious acts. We believe the approach described here is a powerful step in the right direction towards increasing the reliability, security, and survivability of traffic control systems.

Future research should focus in three areas: field testing the system at signalized intersections in a variety of weather conditions; expanding control modifications to include other traffic control parameters, such as passage time, minimum green, and offsets; and increasing the power of the system to maintain reliable, secure, and survivable traffic signal service.

References

American Association of State Highway and Transportation Officials (AASHTO), Institute of Transportation Engineers (ITE), and National Electrical Manufacturers Association (NEMA). NTCIP 1202: NTCIP Object Definitions for Actuated Signal Controller (ASC) Units – version 02. Washington: AASHTO, ITE, and NEMA, November 2005.

Ahmed, S., Abdel-Rahim, A., and Dixon, M., "An External Logic Processor for NTCIP-Based Traffic Controllers: Proof of Concept for Data Exchange Capability", CD-ROM, Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems, St. Louis, MO, USA, October 2009, pp. 779-784.

Al-Kaisy, A. and Z. Freedman. Weather-Responsive Signal Timing: Practical Guidelines. Transportation Research Record 1978, Washington, D.C., 2006, pp.49–60.

Andrey, J., M. Christie, S. Michaels, D.. 2005. Toward a National Assessment of the Travel Risks Associated with Inclement Weather, ISBN 0-9733795-8-8, Institute for Catastrophic Loss Reduction (June).

M. Azadmanesh, A. Krings, and B. Ghahramani, *Global Convergence in Partially Fully Connected Networks (PFCN) with Limited Relays*, International Journal of Information Technology and Decision Making (IJITDM), Vol. 2, No. 2, pp. 265-285, June 2003

M. Azadmanesh, and R. Kieckhafer, *Exploiting Omissive Faults in Synchronous Approximate Agreement*, IEEE Trans. Computers, 49(10), October 2000, pp. 1031-1042.

Bernardin, Lochmueller & Associates. 1995. Anchorage Signal System Upgrade—Final Report.
Maki, P. J. 1999. Adverse Weather Traffic Signal Timing. 69th Annual Meeting of the ITE, Las Vegas, Nevada.

Fambro, Daniel B., Koppa, Rodger J., Picha, Dale L., and Fitzpatrick, Kay. (2000). "Driver Braking Permormance in Stopping Sight Distance Situations." Transportation Research Record, 1701, 9-16.

FHWA (2001). "Surrogate Safety Assessment Model (SSAM)", Final Report, FHWA Publication No.: FHWA-HRT-08-051., Washington, D.C.

FHWA (2009) Available at: http://ops.fhwa.dot.gov/Weather/ <Accessed on May 26, 2009>

Gettman, Douglas; Pu, Lili; Sayed, Tarek; Shelby, Steve (2008). "Surrogate Safety Assessment Model and Validation: Final Report." Turner-Fairbank Highway Research Center, McLean, VA, FHWA Report No. FHWA-HRT-08-051, 14-24.

Gillam, W. and R. Withill. 1992. UTC and Inclement Weather Conditions. Leicestershire County Council and the University of Nottingham in the United Kingdom, presented at the Institute of Electrical and Electronics Engineers Conference, pp. 85–88.

Hranac, Robert; Sterzin, Emily; Krechmer, Daniel; Rakha, Hesham; and Farzaneh, Mohamadreza. (2006). "Empirical Studies on Traffic Flow in Inclement Weather." Cambridge Systematics and Virginia Tech Transportation Institute. FHWA Report No. FHWA-HOP-07-073.

A. Krings, "Survivable Systems", chapter 5, Information Assurance: Dependability and Security in Networked Systems, Morgan Kaufmann Publishers, Yi Qian, James Joshi, David Tipper, and Prashant Krishnamurthy (Editors), 2008.

A. Krings, and T. Feyer, *The Byzantine Agreement Problem: Optimal Early Stopping*, 32nd Hawaii International Conference on System Sciences , No.~stdds03, pp. 1-12, January 5-8, 1999.

A. Krings, W.S. Harrison, N. Hanebutte, C. Taylor, and M. McQueen, *A Two-Layer Approach to Survivability of Networked Computing Systems*, Proc. International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, L'Aquila, Italy, Aug 06 - Aug 12, pp. 1-12, 2001.

L. Lamport, M. Pease, R. Shostak, *The Byzantine Generals Problem*, ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, pp. 382-401.

M. Limber, S. Drobot, and T. Fowler, *Clarus Quality Checking Algorithm Documentation Report,* RITA Intelligent Transportation Systems Joint Program Office, Final Report — December 21, 2010, FHWA-JPO-11-075

Lownes Nicholas E., and Machemehl, Randy B. (2006). "Sensitivity of Simulated Capacity to Modification of VISSIM Driver Behavior Parameters." Transportation Research Record, 1988, 102-110.

Lu, Jian John. (1996). "Vehicle Traction Performance on Snowy and Icy Surfaces." Transportation Research Board, 1536, 82-89.

J. Munson, A. Krings and R. Hiromoto, *The Architecture of a Reliable Software Monitoring System for Embedded Software Systems*, American Nuclear Society 2006 Winter Meeting and Nuclear Technology Expo, 2006.

Nakatsuji, Takashi, and Kawamura, Akira. (2003). "Relationship Between Winter Road-Surface Conditions and Vehicular Motion." Transportation Research Record, 1824, 106-114.

Perrin, J. and P. Martin. 2002. Modifying Signal Timing During Inclement Weather. University of Utah Traffic Lab, Presented at the 2002 Institute of Transportation Engineers Annual Meeting, Washington, D.C.

Pisano P. A. and L. C. Goodwin. 2004. Arterial Operations in Adverse Weather. ITE 2004 Annual Meeting, Institute of Transportation Engineers, Washington, D.C.

Pisano, P. A., L. C. Goodwin and M. A. Rossetti. 2008. U. S. Highway Crashes in Adverse Road Weather Conditions, Presented at the 24th Conference on IIPS, New Orleans, Louisiana.

PTV (2010). “VISSIM User Manual v.5.30”, PTV Planung Transport Verkehr AG, Karlsruhe, Germany.

Seli A. J., S. W. Adel and E. Wael. 2004. Inclement weather and traffic flow at signalized intersections: Case study from northern New England. Transportation Research Record 1867:163–171. Transportation Research Board, National Research Council, Washington, D.C.

Sterzin, Emily D. (2004). “Modeling Influencing Factors in a Microscopic Traffic Simulator.” M.S. thesis, Massachusetts Institute of Technology (MIT), Cambridge, MA.

P. Thambidurai, and Y.-K. Park, *Interactive Consistency with Multiple Failure Modes*, Proc. 7th Symp. on Reliable Distributed Systems, Columbus, OH, Oct. 1988, 93-100.

WTI (2009) Available at: http://www.wti.montana.edu/Projects.aspx?id=89095a90-bbfb-4cb3-a466-ad37b29f34b1 <Accessed on May 28, 2009>

Ye, Z. 2009. Evaluation of the Utah DOT Weather Operations/RWIS Program on Traffic Operations. Draft Final Report. Prepared for the Iowa Department of Transportation and the Aurora Program, April 2009.

Notice

Unless otherwise mentioned in the Figure/Table caption, all Figures, Tables and Photos in this document were made, developed and produced by the University of Idaho’s National Institute for Advanced Transportation Technology faculty, staff, and students and are used here with permission. The photo used in the cover page by Ahmed Abdel-Rahim for Main Street in downtown Moscow Idaho.

APPENDIX A. List of Acronyms

ASC	Actuated Signal Controllers
CID	Controller Interface Device
ESS	Environmental Sensor Stations
FHWA	Federal Highway Administration
HILS	Hardware-in-the-loop Simulation
ITS	Intelligent Transportation Systems
NTCIP	National Transportation Communications for ITS Protocol
PET	Post-encroachment Time
RWIS	Road Weather Information System
SSAM	Surrogate Safety Assessment Model
SNMP	Simple Network Management Protocol
STMP	Simple Transportation Management Protocol
TRJ	Trajectory File
TTC	Time-to-collision
vphpl	Vehicles Per Hour Per Lane

APPENDIX B. NTCIP 1202 Object Access Status

Table B-1. NTCIP 1202 Object Access Status

NTCIP Clause	Accessibility Level
2.2 PHASE PARAMETERS	
2.2.1 Maximum Phases	Read-only
2.2.2 Phase Table	Not-accessible
2.2.2.1 Phase Number	Read-only
2.2.2.2 Phase Walk Parameter	Read-write
2.2.2.3 Phase Pedestrian Clear Parameter	Read-write
2.2.2.4 Phase Minimum Green Parameter	Read-write
2.2.2.5 Phase Passage Parameter	Read-write
2.2.2.6 Phase Maximum Green 1 Parameter	Read-write
2.2.2.7 Phase Maximum Green 2 Parameter	Read-write
2.2.2.8 Phase Yellow Change Parameter	Read-write
2.2.2.9 Phase Red Clear Parameter	Read-write
2.2.2.10 Phase Red Revert	Read-write
2.2.2.11 Phase Added Initial Parameter	Read-write
2.2.2.12 Phase Maximum Initial Parameter	Read-write
2.2.2.13 Phase Time Before Reduction Parameter	Read-write
2.2.2.14 Phase Cars Before Reduction Parameter	Read-write
2.2.2.15 Phase Time To Reduce Parameter	Read-write
2.2.2.16 Phase Reduce By	Read-write
2.2.2.17 Phase Minimum Gap Parameter	Read-write
2.2.2.18 Phase Dynamic Max Limit	Read-write
2.2.2.19 Phase Dynamic Max Step	Read-write
2.2.2.20 Phase Startup	Read-write
2.2.2.21 Phase Options	Read-write
2.2.2.22 Phase Ring Parameter	Read-write
2.2.2.23 Phase Concurrency	Read-write
2.2.3 Maximum Phase Groups	Read-only
2.2.4 Phase Status Group Table	Not accessible
2.2.4.1 Phase Status Group Number	Read-only
2.2.4.2 Phase Status Group Reds	Read-only
2.2.4.3 Phase Status Group Yellows	Read-only
2.2.4.4 Phase Status Group Greens	Read-only
2.2.4.5 Phase Status Group Dont Walks	Read-only
2.2.4.6 Phase Status Group Pedestrian clears	Read-only
2.2.4.7 Phase Status Group Walks	Read-only
2.2.4.8 Phase Status Group Vehicle Calls	Read-only
2.2.4.9 Phase Status Group Pedestrian Calls	Read-only
2.2.4.10 Phase Status Group Phase Ons	Read-only
2.2.4.11 Phase Status Group Phase Nexts	Read-only

Table B-1. NTCIP 1202 Object Access Status (Cont.)

NTCIP Clause	Accessibility Level
2.2.5 Phase Control Table	Not accessible
2.2.5.1 Phase Control Group Number	Read-only
2.2.5.2 Phase Omit Control	Read-write
2.2.5.3 Pedestrian Omit Control	Read-write
2.2.5.4 Phase Hold Control	Read-write
2.2.5.5 Phase Force Off Control	Read-write
2.2.5.6 Vehicle Call Control	Read-write
2.2.5.7 Pedestrian Call Control	Read-write
2.3 DETECTOR PARAMETERS	
2.3.1 Maximum Vehicle Detectors	Read-only
2.3.2 Vehicle Detector Parameter Table	Not accessible
2.3.2.1 Vehicle Detector Number	Read-only
2.3.2.2 Vehicle Detector Options Parameter	Read-write
2.3.2.3 Vehicle Detector Call Phase Parameter	Read-write
2.3.2.4 Vehicle Detector Switch Phase Parameter	Read-write
2.3.2.5 Vehicle Detector Delay Parameter	Read-write
2.3.2.6 Vehicle Detector Extend Parameter	Read-write
2.3.2.7 Vehicle Detector Queue Limit	Read-write
2.3.2.8 Vehicle Detector No Activity Parameter	Read-write
2.3.2.9 Vehicle Detector Maximum Presence Parameter	Read-write
2.3.2.10 Vehicle Detector Erratic Counts Parameter	Read-write
2.3.2.11 Vehicle Detector Fail Time Parameter	Read-write
2.3.2.12 Vehicle Detector Alarms	Read-only
2.3.2.13 Vehicle Detector Reported Alarms	Read-only
2.3.2.14 Vehicle Detector Reset	Read-write
2.3.3 Maximum Vehicle Detector Status Groups	Read-only
2.3.4 Vehicle Detector Status Group Table	Not accessible
2.3.4.1 Detector Status Group Number	Read-only
2.3.4.2 Detector Status Group Active	Read-only
2.3.4.3 Detector Alarm Status	Read-only
2.3.5 Volume / Occupancy report	Read-only
2.3.5.1 Volume / Occupancy Sequence	Read-only
2.3.5.2 Volume / Occupancy Period	Read-write
2.3.5.3 Active Volume / Occupancy Detectors	Read-only
2.3.5.4 Volume / Occupancy Table	Not accessible
2.3.5.4.1 Volume data	Read-only
2.3.5.4.2 Occupancy data	Read-only
2.3.6 Maximum Pedestrian Detectors	Read-only

Table B-1. NTCIP 1202 Object Access Status (Cont.)

NTCIP Clause	Accessibility Level
2.3.7 Pedestrian Detector Parameter Table	Not accessible
2.3.7.1 Pedestrian Detector Number	Read-only
2.3.7.2 Pedestrian Detector Call Phase Parameter	Read-write
2.3.7.3 Pedestrian Detector No Activity Parameter	Read-write
2.3.7.4 Pedestrian Detector Maximum Presence Parameter	Read-write
2.3.7.5 Pedestrian Detector Erratic Counts Parameter	Read-write
2.3.7.6 Pedestrian Detector Alarms	Read-only
2.4 UNIT PARAMETERS	
2.4.1 StartUp Flash Parameter read-write	Read-write
2.4.2 Automatic Ped Clear Parameter	Read-write
2.4.3 Backup Time Parameter	Read-write
2.4.4 Unit Red Revert Parameter	Read-write
2.4.5 Unit Control Status	Read-only
2.4.6 Unit Flash Status	Read-only
2.4.7 Unit Alarm Status 2	Read-only
2.4.8 Unit Alarm Status 1	Read-only
2.4.9 Short Alarm Status	Read-only
2.4.10 Unit Control	Read-write
2.4.11 Maximum Alarm Groups	Read-only
2.4.12 Alarm Group Table	Not-accessible
2.4.12.1 Alarm Group Number	Read-only
2.4.12.2 Alarm Group State	Read-only
2.4.13 Maximum Special Function Outputs	Read-only
2.4.14 Special Function Output Table	Not-accessible
2.4.14.1 Special Function Output Number	Read-write
2.4.14.2 Special Function Output Control	Read-write
2.5 COORDINATION PARAMETERS	
2.5.1 Coord Operational Mode Parameter	Read-write
2.5.2 Coord Correction Mode Parameters	Read-write
2.5.3 Coord Maximum Mode Parameters	Read-write
2.5.4 Coord Force Mode Parameters	Read-write
2.5.5 Maximum Patterns Parameters	Read-only
2.5.6 Pattern Table Type	Read-only
2.5.7 Pattern Table	Not-accessible
2.5.7.1 Pattern Number Entry	Read-only
2.5.7.2 Pattern Cycle Time	Read-write
2.5.7.3 Pattern Offset Time Parameter	Read-write
2.5.7.4 Pattern Split Number Parameter	Read-write
2.5.7.5 Pattern Sequence Number Parameter	Read-write
2.5.8 Maximum Splits	Read-only

Table B-1. NTCIP 1202 Object Access Status (Cont.)

NTCIP Clause	Accessibility Level
2.5.9 Split Table	Not-accessible
2.5.9.1 Split Number	Read-only
2.5.9.2 Split Phase Number	Read-only
2.5.9.3 Split Time Parameter	Read-write
2.5.9.5 Split Coordinated Phase	Read-write
2.5.10 Coordination Pattern Status	Read-only
2.5.11 Local Free Status	Read-only
2.5.12 Coordination Cycle Status	Read-only
2.5.13 Coordination Sync Status	Read-only
2.5.14 System Pattern Control	Read-write
2.5.15 System Sync Control	Read-write
2.6 TIME BASE PARAMETERS	
2.6.1 Time Base Pattern Sync Parameter	Read-write
2.6.2 Maximum Time Base Actions	Read-only
2.6.3 Time Base ASC Action Table	Not-accessible
2.6.3.1 Time Base Action Number Entry	Read-only
2.6.3.2 Time Base Action Pattern Parameter	Read-write
2.6.3.3 Time Base Action Auxiliary Function Parameter	Read-write
2.6.3.4 Time Base Action Special Function Parameter	Read-write
2.6.4 Time Base ASC Action Status	Read-only
2.7 PREEMPT PARAMETERS	
2.7.1 Maximum Preempts	Read-only
2.7.2 Preempt Table	Not-accessible
2.7.2.1 Preempt Number	Read-only
2.7.2.2 Preempt Control Parameter	Read-write
2.7.2.3 Preempt Link Parameter	Read-write
2.7.2.4 Preempt Delay Parameter	Read-write
2.7.2.5 Preempt Duration Parameter	Read-write
2.7.2.6 Preempt Minimum Green Parameter	Read-write
2.7.2.7 Preempt Minimum Walk Parameter	Read-write
2.7.2.8 Preempt Enter Pedestrian Clear Parameter	Read-write
2.7.2.9 Preempt Track Green Parameter	Read-write
2.7.2.10 Preempt Minimum Dwell Parameter	Read-write
2.7.2.11 Preempt Maximum Presence Parameter	Read-write
2.7.2.12 Preempt Track Phase Parameter	Read-write
2.7.2.13 Preempt Dwell Phase Parameters	Read-write
2.7.2.14 Preempt Dwell Ped Parameters	Read-write
2.7.2.15 Preempt Exit Phase Parameters	Read-write
2.7.2.16 Preempt State	Read-only

Table B-1. NTCIP 1202 Object Access Status (Cont.)

NTCIP Clause	Accessibility Level
2.7.3 Preempt Control Table	Not-accessible
2.7.3.1 Preempt Control Number	Read-only
2.7.3.2 Preempt Control State	Read-only
2.8 RING PARAMETERS	
2.8.1 Maximum Rings	Read-only
2.8.2 Maximum Sequences	Read-only
2.8.3 Sequence Table	Not-accessible
2.8.3.1 Sequence Number	Read-only
2.8.3.2 Sequence Ring Number	Read-write
2.8.3.3 SequenceData read-write	Read-write
2.8.4 Maximum Ring Control Groups	Read-only
2.8.5 Ring Control Group Table	Not-accessible
2.8.5.1 Ring Control Group Number	Read-only
2.8.5.2 Ring Stop Time Control	Read-write
2.8.5.3 Ring Force Off Control	Read-write
2.8.5.4 Ring Max 2 Control	Read-write
2.8.5.5 Ring Max Inhibit Control	Read-write
2.8.5.6 Ring Ped Recycle Control	Read-write
2.8.5.7 Ring Red Rest Control	Read-write
2.8.5.8 Ring Omit Red Control	Read-write
2.9 CHANNEL PARAMETERS	
2.9.1 Maximum Channels	Read-only
2.9.2 Channel Table	Not-accessible
2.9.2.1 Channel Number	Read-only
2.9.2.2 Channel Control Source Parameters	Read-only
2.9.2.3 Channel Control Type Parameters	Read-only
2.9.2.4 Channel Flash Parameters	Read-only
2.9.2.5 Channel Dim Parameters	Read-only
2.9.3 Maximum Channel Status Groups	Read-only
2.9.4 Channel Status Group Table	Not-accessible
2.9.4.1 Channel Status Group Number	Read-only
2.9.4.2 Channel Status Group Reds	Read-only
2.9.4.3 Channel Status Group Yellows	Read-only
2.9.4.4 Channel Status Group Greens	Read-only

Table B-1. NTCIP 1202 Object Access Status (Cont.)

NTCIP Clause	Accessibility Level
2.10 OVERLAP PARAMETERS	
2.10.1 Maximum Overlaps	Read-only
2.10.2 Overlap Table	Not-accessible
2.10.2.1 Overlap Number	Read-only
2.10.2.2 Overlap Type	Read-write
2.10.2.3 Overlap Included Phase Parameters	Read-write
2.10.2.4 Overlap Modifier Phase Parameters	Read-write
2.10.2.5 Overlap Trailing Green Parameter	Read-write
2.10.2.6 Overlap Trailing Yellow Change Parameter	Read-write
2.10.2.7 Overlap Trailing Red Clear Parameter	Read-write
2.10.3 Maximum Overlap Status Groups	Read-only
2.10.4 Overlap Status Group Table	Not-accessible
2.10.4.1 Overlap Status Group Number	Read-only
2.10.4.2 Overlap Status Group Reds	Read-only
2.10.4.3 Overlap Status Group Yellows	Read-only
2.10.4.4 Overlap Status Group Greens	Read-only
2.11 TS2 PORT 1 PARAMETERS	
2.11.1 Maximum Port 1 Addresses	Read-only
2.11.2 Port 1 Table	Not-accessible
2.11.2.1 Port 1 Number	Read-only
2.11.2.2 Port 1 Device Present	Read-only
2.11.2.3 Port 1 Frame 40 Enable	Read-only
2.11.2.4 Port 1 Status	Read-write
2.11.2.5 Port 1 Fault Frame	Read-only

U.S. Department of Transportation
ITS Joint Program Office-HOIT
1200 New Jersey Avenue, SE
Washington, DC 20590

Toll-Free “Help Line” 866-367-7487
www.its.dot.gov

[FHWA-JPO-12-016]

U.S. Department of Transportation
Federal Highway Administration
Research and Innovative Technology Administration