IDAHO TRANSPORTATION DEPARTMENT

RESEARCH REPORT

RP 180

An Automated Testing Tool For Traffic Signal Controller Functionalities

By

Sk. Monsur Ahmed; Cody Browne,

Ahmed Abdel-Rahim, Ph.D.; Richard Wall, Ph.D.

National Institute for Advanced Transportation Technology

University of Idaho

Prepared for

Idaho Transportation Department

Research Section

Transportation Planning Division

http://itd.idaho.gov/planning/research/

March 2010

1. Report No. FHWA-ID-10-180	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle An Automated Testing Tool for Traffic Signal Controller Functionalities	5. Report Date March 2010 6. Performing Organization Code
7. Author(s) Ahmed Abdel-Rahim; Richard Wall; Sk. Monsur Ahmed; Cody Browne	8. Performing Organization Report No.
9. Performing Organization Name and Address National Institute for Advanced Transportation Technology University of Idaho PO Box 440901; 115 Engineering Physics Building Moscow, ID 838440901	10. Work Unit No. (TRAIS) 11. Contract or Grant No. RP180
12. Sponsoring Agency Name and Address Idaho Transportation Department PO Box 7129 Boise, ID 83707-7129	13. Type of Report and Period Covered Final Report January 2008 – March 2010 14. Sponsoring Agency Code

15. Supplementary Notes:

16. Abstract

The purpose of this project was to develop an automated tool that facilitates testing of traffic controller functionality using controller interface device (CID) technology. Benefits of such automated testers to traffic engineers include reduced testing time, enhanced repeatability and consistency of testing, reduced testing costs, and improved testing quality and productivity. The automated tester can be operated in a static mode using the graphical user interface. The timing of input changes is strictly controlled by the person operating the testing system. It uses XML script files to specify which inputs are activated, the timing of those activations and verifying the controller response(s). The software provided is for a limited set of NEMA TS1 controllers running a specific firmware version as specified by the Idaho Transportation Department. Other traffic controllers can be tested provided that the testing program has been modified to communicate with that specific controller. Software modifications to the testing software are required because there is no standard communications protocol used by various traffic controller manufacturers that allow the traffic controller settings to be read from the controller. A version of the automated tester has been developed and tested that interfaces with NEMA TS2 type 1 and type 2 controllers. This version uses the National Transportation Communications for ITS Protocol (NTCIP) via either the asynchronous RS232 serial port or the Ethernet port. However, our investigation has shown that various vendors have significant differences in the interpretation of the NTCIP standard, and the automated testing software must still be verified with specific traffic devices running specific firmware versions. The automated tester also includes a CID-based suitcase tester emulator that can replace or supplement manual testing.

17. Key Words	18. Distribution Statement
traffic signal controllers; testing; automatic test equipment; traffic signal timing; fuzzy logic	Unrestricted; Copies available online from http://itd.idaho.gov/planning/research/archived/closed.htm and http://www.webs1.uidaho.edu/niatt/research.asp

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	29	

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**

SI* (MODERN METRIC) CONVERSION FACTORS

APPROXIMATE CONVERSIONS TO SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
		LENGTH		
in	inches	25.4	millimeters	mm
ft	feet	0.305	meters	m
yd	yards	0.914	meters	m
mi	miles	1.61	kilometers	km
		AREA		
in^2	square inches	645.2	square millimeters	mm^2
ft^2	square feet	0.093	square meters	m^2
yd^2	square yard	0.836	square meters	m^2
ac	acres	0.405	hectares	ha
mi^2	square miles	2.59	square kilometers	km^2
		VOLUME		
fl oz	fluid ounces	29.57	milliliters	mL
gal	gallons	3.785	liters	L
ft^3	cubic feet	0.028	cubic meters	m^3
yd^3	cubic yards	0.765	cubic meters	m^3
		NOTE: volumes greater than 1000 L shall be shown in m^3		
		MASS		
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2000 lb)	0.907	megagrams (or "metric ton")	Mg (or "t")
		TEMPERATURE (exact degrees)		
°F	Fahrenheit	5 (F-32)/9 or (F-32)/1.8	Celsius	°C
		ILLUMINATION		
fc	foot-candles	10.76	lux	lx
fl	foot-Lamberts	3.426	candela/m^2	cd/m^2
		FORCE and PRESSURE or STRESS		
lbf	poundforce	4.45	newtons	N
lbf/in^2	poundforce per square inch	6.89	kilopascals	kPa

APPROXIMATE CONVERSIONS FROM SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
		LENGTH		
mm	millimeters	0.039	inches	in
m	meters	3.28	feet	ft
m	meters	1.09	yards	yd
km	kilometers	0.621	miles	mi
		AREA		
mm^2	square millimeters	0.0016	square inches	in^2
m^2	square meters	10.764	square feet	ft^2
m^2	square meters	1.195	square yards	yd^2
ha	hectares	2.47	acres	ac
km^2	square kilometers	0.386	square miles	mi^2
		VOLUME		
mL	milliliters	0.034	fluid ounces	fl oz
L	liters	0.264	gallons	gal
m^3	cubic meters	35.314	cubic feet	ft^3
m^3	cubic meters	1.307	cubic yards	yd^3
		MASS		
g	grams	0.035	ounces	oz
kg	kilograms	2.202	pounds	lb
Mg (or "t")	megagrams (or "metric ton")	1.103	short tons (2000 lb)	T
		TEMPERATURE (exact degrees)		
°C	Celsius	1.8C+32	Fahrenheit	°F
		ILLUMINATION		
lx	lux	0.0929	foot-candles	fc
cd/m^2	candela/m^2	0.2919	foot-Lamberts	fl
		FORCE and PRESSURE or STRESS		
N	newtons	0.225	poundforce	lbf
kPa	kilopascals	0.145	poundforce per square inch	lbf/in^2

*SI is the symbol for the International System of Units. Appropriate rounding should be made to comply with Section 4 of ASTM E380. (Revised March 2003)

Table of Contents

List of Tables

List of Figures

Abstract

The purpose of this project was to develop an automated tool that facilitates testing of traffic controller functionality using controller interface device (CID) technology. Benefits of such automated testers to traffic engineers include reduced testing time, enhanced repeatability and consistency of testing, reduced testing costs, and improved testing quality and productivity. The automated tester can be operated in a static mode using the graphical user interface. The timing of input changes is strictly controlled by the person operating the testing system. It uses XML script files to specify which inputs are activated, the timing of those activations and verifying the controller response(s). The software provided is for a limited set of NEMA TS1 controllers running a specific firmware version as specified by the Idaho Transportation Department. Other traffic controllers can be tested provided that the testing program has been modified to communicate with that specific controller. Software modifications to the testing software are required because there is no standard communications protocol used by various traffic controller manufacturers that allow the traffic controller settings to be read from the controller. A version of the automated tester has been developed and tested that interfaces with NEMA TS2 type 1 and type 2 controllers. This version uses the National Transportation Communications for ITS Protocol (NTCIP) via either the asynchronous RS232 serial port or the Ethernet port. However, our investigation has shown that various vendors have significant differences in the interpretation of the NTCIP standard, and the automated testing software must still be verified with specific traffic devices running specific firmware versions. The automated tester also includes a CID-based suitcase tester emulator that can replace or supplement manual testing.

Chapter 1
Introduction

Background

The State of Idaho's Intelligent Transportation System (ITS) strategic plan identifies several traffic signal improvement projects as short term high-priority projects.[1] Updating traffic controllers and cabinets are major components of these signal improvement projects. Idaho Transportation Department (ITD) traffic engineers and technicians are faced with the challenge of testing these new controllers to ensure that they comply with the state requirements prior to deployment. Currently, the test is being done manually using "suitcase testers" that require the user to input commands manually. The information related to any test cannot be automatically recorded or documented. The user is required to manually execute the test commands and input the sequence of instructions each time the test is to be repeated. This process can be both tedious and time consuming and is vulnerable to human errors in all steps of the testing procedures.

The purpose of this research project was to develop and test an automated testing tool capable of testing different traffic controller functionalities to replace or supplement manual testing. Benefits of such automated testers to traffic engineers include reduced testing time, enhanced repeatability and consistency of testing, reduced testing costs, and improved testing quality and productivity. An automated testing tool will also allow ITD to implement a unified testing procedure for the traffic sections in all districts.

The automated testing tool developed in this project utilizes the University of Idaho's National Institute for Advanced Transportation Technology (NIATT) Controller Interface Device (CID).[2] As a result of a recent ITD research project, all six ITD districts have a CID that can be utilized as part of the testing tool. ITD District traffic engineers are familiar with the CID operation and capabilities, which will make it easier to integrate it within the automated testing procedure. Our plan is to generate computer models that are "trained" to recognize traffic controllers that meet performance expectations using fuzzy logic system identification techniques. The research had the following four objectives:

1. Develop and validate an automated testing tool for traffic signal controller functionality.
2. Automate the testing processes and standardize the test observations to reduce operator involvement.
3. Improve the quality of testing without increasing expense.
4. Develop guidelines and training materials that will assist ITD traffic engineers and technicians in using the automated testing tool.

Project Description

The project had eight major tasks. Investigators were to:

1. Identify advisory group and technical oversight committee for the project.
   The advisory group consists of producers, users, and researchers to direct the focus and implementation for the tester.

2. Characterize traffic controller operating parameters.
   a. Classify controller operations to rank performance.
   b. Develop assessment metrics for performance and capability of traffic controllers.
   c. Identify common failure modes from the experiences of users.

3. Develop excitation models and operational characteristics of traffic controllers.
   a. Develop mapping algorithms for translating common descriptions of signal operations to computer executable instructions.
   b. Develop a strategy for accessing the traffic controller operations database.

4. Test models for degree of fault coverage and degree of fault observation or differentiation.
   a. Classify failure modes.
   b. Develop fault tree for hardware failures.
   c. Develop signal timing windows and tolerances.

5. Develop specifications for practices, procedures, and interfaces to complete traffic controller validation.

6. Implement preliminary hardware for proof of concept verification.
   a. Integrate PC with CID and traffic controller.
   b. Run beta testing with a standard signal timing plan developed by the project team.
   c. Induce specific failures and document results as part of the development validation.
   d. Review automated tester performance with advisory group.

7. Conduct training workshop for ITD traffic engineers and technicians.

8. Publish results.
   a. Report on system performance to ITD.
   b. Publish a user's manual.
   c. Publish technical description in technical and professional society proceedings.

Report Organization

This report is organized in three sections. After the introduction, Section 2 introduces the CID-based suitcase tester emulator and its functions for both NEMA-TS1 and NEMA-TS2 type controllers. Section 3 documents the characteristics of the automated tester covering its hardware and software requirements, architecture, and data flow. In addition to this report, the project deliverables include the automated tester installation program and its complete users' manual.

Chapter 2
CID-Based Suitcase Tester Emulator

Overview

Traffic engineers have traditionally used a suitcase tester to test and evaluate the operation of traffic controllers. A suitcase tester is a suitcase-like device containing switches and LEDs that, when linked to a traffic controller activates different controller functions and monitors outputs from the controller. A suitcase tester provides a controlled environment for verifying that the controller operates as expected when individual functions are programmed. Traditional suitcase tester usage can be time-consuming and inefficient, particularly when it is necessary to verify the operation of all of the controller's functions. In addition, each type of traffic controller on the market requires its own suitcase tester.

In recent years, with the emergence of CID technology in the traffic industry, several suitcase tester emulator software packages were developed to supplement shortcomings of the traditional suitcase tester (for example, the Siemens Intelligent Transportation Systems and the NAZTEC NEMA-TS2 testers).[3, 4] These NEMA-TS2 testers consist of two main parts: the tester device and the emulator software. The SDLC port is used to communicate with the traffic controller and the COM port is used to connect with the computer that runs the emulator software. These test box emulators, however, come with certain limitations. The number of traffic controllers that they can test simultaneously is also limited. In addition, a pulse detector signal cannot be generated as the length of a detector call cannot be specified less than the time step at which the software and controller are communicating.

The suitcase tester emulator used in this project utilizes the CID suitcase tester emulator software developed at NIATT. [5] The software is a computerized version of the traditional suitcase tester. Just like the traditional suitcase tester, the suitcase tester emulator can verify that the controller has been programmed correctly and troubleshoots problems with the controller. It goes beyond traditional suitcase testers in that it can test the most commonly used functions of any NEMA or type-170 controllers, rather than those of just one specific model. It can also test multiple controllers at the same time.

The suitcase tester emulator implements three user interfaces, for use with three different controller types – NEMA-TS1, NEMA-TS2 (type-2), and type-170.[6,7,8] Each interface includes two basic functions – input and output. The input functions are used to activate the controller's input functions. The output indications are used to monitor the traffic controller's outputs. Figure 1 shows the suitcase tester screen of a NEMA TS1 controller.

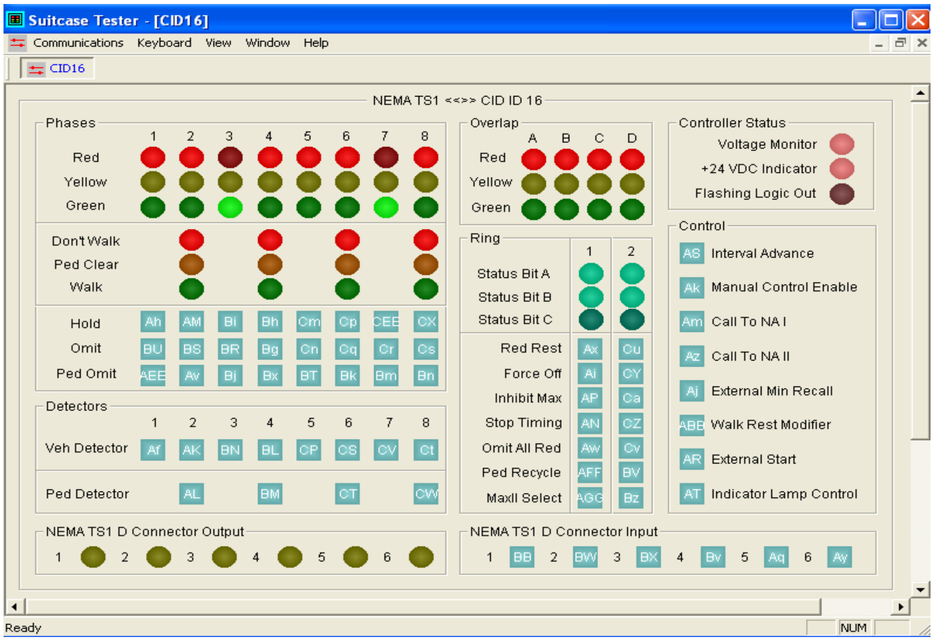


Figure 1: Suitcase Tester Screen of a NEMA TS1 Type Controller.

The output buttons are used to monitor the traffic controller output functions (such as phases). These output buttons emulate the traditional suitcase tester device's LEDs. The input buttons are used to activate traffic controller input functions such as detector calls. The input buttons emulate the traditional suitcase tester device's mechanical switches, but with more functions than the mechanical switches are able to provide. In addition, the input buttons in the NIATT CID suitcase tester emulator generate a variety of possible types of signal patterns that could occur in the field. The three possible signal types are random, presence, and deterministic.

- Random signal: The random signal begins at a high-voltage (24-V for a NEMA controller or 12-V for a type-170 controller) signal, then cycles to a zero-voltage signal, then returns to the high voltage signal, and so on. The duration of each high-voltage signal and each zero-voltage signal is random, as are the intervals between signals. This signal pattern represents vehicles arriving at an intersection randomly.

- Presence signal: The presence signal begins as the high-voltage signal, cycles to a zero-voltage signal, and remains there until the user deactivates it. There are no variations in this type of signal. This signal pattern represents the vehicles stopping at intersection waiting for the traffic signal to change to green.

- Deterministic signal: The deterministic signal begins as a zero-voltage signal, cycles to the high-voltage signal, and then returns to zero again. In contrast to the random signal, the user can specify the duration of the high-voltage pulse and the interval between pulses. This signal type will repeat the same pattern until it is stopped. This signal pattern represents vehicles passing through the intersection at a constant flow rate.

Hardware Testing: NEMA TS1 Controllers

The following features are available under the “control menu” section on a suitcase tester in a NEMA TS1 controller.

Interval Advance

The interval advance button is used for stepwise skipping of phases in a controller. When the interval advance is used, a phase which is initially green turns yellow. With further use of the interval advance function, the signal turns red and moves to the next phase.

Expected Results

The interval advance control should be able to move the signal phase when invoked. If a controller fails to respond to this function, then there is a malfunction in the controller.

Manual Control Enable

The manual control enable is a function that should be input before any manual input operation can be made into the controller.

Expected Results

The manual control button should be enabled before any manual inputs can be made to the controller unit during the testing procedure. If this procedure does not function as expected, the controller logic needs to be checked for functionality.

Call to NA I

The call to non-actuated I button is used to prevent the controller unit from operating in a coordinated mode. For the controller unit to be able to operate in a non-actuated mode when this feature is activated, the desired phases need to be selected in the controller options section on the controller unit.

Expected Results

The call to non-actuated I function is expected to put the phase in a non-actuated mode. If this function acts otherwise, the following are the probable causes:

- The status of the phase in the controller options under the configuration menu is wrong.
- The manual control enable is not selected.
- The minimum green time is more than the maximum green time.

Call to NA II

The call to non-actuated II functions the same way as the call to non-actuated I except that this feature is performed on ring two of a standard NEMA plan.

Expected results

The call to non-actuated II functions in the same manner as the call to non-actuated I.

External Minimum Recall

When used, the external minimum function causes all phases to reoccur and to serve a minimum amount of green time.

Walk Rest Modifier

When the walk rest modifier is used, it makes non-actuated phases remain in the “rest in walk” state in the absence of serviceable or conflicting calls.

External Start

External start reinitializes the controller into start up like a power outage.

Testing Phases Input and Output Channels

The suitcase tester can test the input and output channels for standard NEMA eight–phase operations. It tests the red, yellow, green, pedestrian don’t walk, pedestrian clearance, and walk signal indications as presented in Figure 1. The suitcase tester also tests the hold, omit, and pedestrian omit functions.

Hold

The hold input is used to manually put a phase on hold in the controller and thus prevent the particular phase from being serviced. If the phase is serviced and held, it will continue being served and other phases will be serviced.

Expected Results

When the hold function is invoked for a phase, the phase is expected to stay in its current signal display. If the phase changes its signal indication while the hold function is still active then,

The manual enable control is not activated, or
The controller is malfunctioning.

Omit

The omit input is used to manually omit a phase from being serviced by the controller.

Expected Results

When the omit function is activated the selected phase is expected not to time during the cycle timing sequence. However, if the selected phase runs while the "omit" function is still activated then, either:

The manual enable control is not activated, or
There is a malfunction in the controller.

Pedestrian Omit

The pedestrian omit (ped omit) button is used to manually omit pedestrian phases from being serviced by the controller.

Expected Results

When the "ped omit" function is activated, the selected pedestrian phase is supposed to be exempted from timing during the cycle. If the selected pedestrian phase runs during the cycle while the pedestrian omit function has been activated, the probable causes are either

The manual enable control is not activated, or
A malfunction in the controller.

Testing Detectors Input Channels

There are two sets of detector inputs on the suitcase tester: vehicle detector and the pedestrian detector.

Vehicle Detector

The eight vehicle detector inputs on the suitcase tester are used to manually simulate the presence of a vehicle at the intersection.

Expected Results

When a vehicle detector input is activated, the controller is expected to respond to the call and serve the phase in the earliest possible sequence according to the sequence of phases and timing plan. However, if the controller does not respond to the vehicle detector call, there is a failure which can be rectified by

Checking to ensure that there are detectors placed in the network and have been activated. The manual control enable function is active.

Pedestrian Detector

Four pedestrian detector inputs on the suitcase tester are used to send signals to the controller that a pedestrian is present at the intersection.

Expected Results

When the pedestrian detector functions when activated in a simulation, the controller is expected to respond to and provide service to the pedestrian phase where the call has been placed. If, however, the controller does not respond to the call, then there is a failure. This failure could result from any one or combination of the following:

The absence of pedestrian detectors in the network.
Inactivated pedestrian detectors.
Malfunctioning controller unit.

Testing Ring Structure

The suitcase tester displays the status of the controller during every signal change. The status is displayed for each ring under status bit A, status bit B or status bit C as presented in Figure 1.

Red Rest

The red rest input button is used to manually request for a red rest in the controller unit and in the selected ring.

Force Off

The force off function is used to manually terminate a phase in the controller unit and in the particular ring.

Expected Results

If the “force off” function does not terminate a phase as expected, then there is a failure in the controller.

Inhibit Max

The inhibit max function is used to manually request for the maximum timing in the controller device to be inhibited.

Expected Results

The inhibit max function is expected to prevent maximum timing in the controller unit. If, however, this function does not prevent the maximum timing, then there is a failure in the controller. This failure is due to either

> The manual control enable not activated.
> A malfunctioning controller unit.

Stop Timing

The stop timing feature is used to manually request the timing of a phase to stop.

Expected results

The stop timing function when invoked is expected to stop the controller from proceeding with its timing sequence.

Omit All Red

The “omit all red” feature is used to manually omit the red clearance interval for a particular phase in the selected ring.

Expected results

The “omit all red” feature is expected to make a phase time without the red clearance interval. If a phase time includes the red clearance time when this function has been invoked, then the possible causes of this failure are

> The manual control enable was not selected prior to using the “omit all red’ for the particular phase.
> There is breakdown in the controller logic.

Pedestrian Recycle

The pedestrian recycle is used to request for a recycle of the pedestrian phase timings in the controller device for the selected ring.

Expected Results

If pedestrian recycle function fails to perform a recycling of the phase as it should, then the possible cause of this failure is

> There is no active pedestrian phase in the timing sequence for the particular ring or there is not enough time to re-service it in the split.

Max II Select

The “max II” function is used to manually request for maximum II timing values in the controller device.

Expected results

When the “max II” function is used, the controller is expected to use the value of the “max II” time for the maximum green or else there is a failure in the controller.

Testing Overlap

The overlap feature on the suitcase tester screen shows the status of the communication port at any point during the operation of the CID for overlap A, B, C, and D.

Hardware Testing: NEMA TS2 Controllers

The suitcase tester emulator for NEMA-TS2 Type controller has three different input/output (I/O) modes of operations: NEMA-TS2 mode 0, NEMA-TS2 mode 1, and NEMA-TS2 mode 2. The suitcase tester emulator for NEMA-TS2 type controllers come with all the test features included in NEMA-TS1 suitcase tester emulator with additional features for I/O mode, preempts, pedestrian phase control, and vehicle phase control that varies depending on the mode of operations. Figure 2 shows the suitcase tester screen for a NEMA-TS2 type controller. The I/O buttons in the lower right side of the screen are used to select the I/O mode of operations. The suitcase tester for NEMA-TS2 Mode 0 is similar to the NEMA-TS1 suitcase tester.

Figure 2: Suitcase Tester Screen of a NEMA TS2 Type Controller.

Additional Inputs for TS2 Controllers in Mode 1

Additional input for this mode include: six preempt inputs, three offset inputs, four timing plan inputs, eight additional detector inputs, four alternative sequence control inputs, and three general control inputs as can be seen in Figure 3.

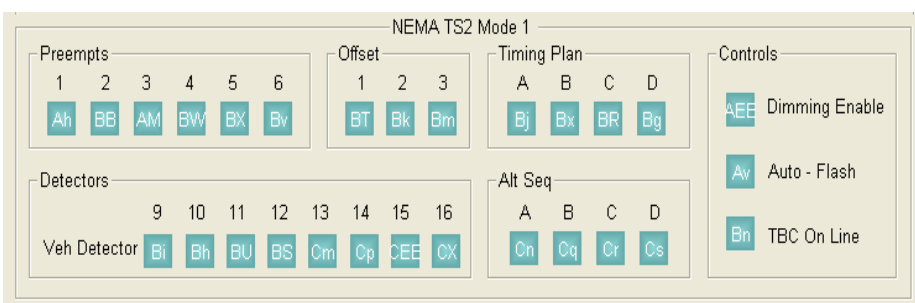
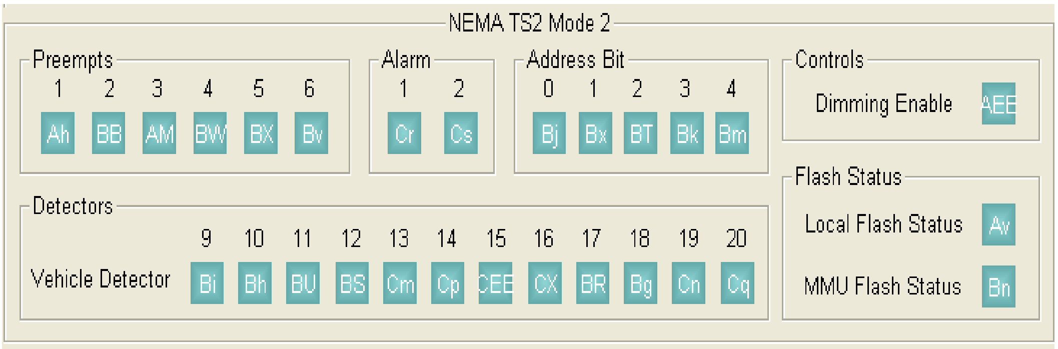


Figure 3: Additional Inputs on a NEMA TS2 Controller in Mode 1.

Additional Inputs for TS2 Controllers in Mode 2

Additional input for this mode include 6 preempt inputs, 2 alarm inputs, 12 additional detector inputs, 4 address bit inputs, a dimming enable control input, and 2 flash status inputs. These additional inputs are shown in Figure 4.



.

Figure 4: Additional Inputs on a NEMA TS2 Controller in Mode 2.

Chapter 3
Testing Controller Functionality: Automated Tester

Introduction and Overview

Traffic controllers control the intersection to provide safe movement of vehicle and pedestrian traffic. To ensure safety, the controller eliminates the conflict points between vehicular and pedestrian movements. Properly operating controllers are loaded with many functions, and failure of any of these functions could cause the operation to be unsafe for the users. Operation of controller functions should be tested and monitored frequently.

System testing of software and/or hardware is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limiting type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing is an investigatory testing phase, where the focus is to have almost a destructive attitude and tests not only the design, but also the behavior and even the believed expectations of the user. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s). Test automation is the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions. Commonly, test automation involves automating a manual process already in place that uses a formalized testing process.

Although manual testing can find many defects in a software based application, it is a laborious and time consuming process. In addition, it may not be effective in finding certain classes of defects. Test automation is a process of writing a computer program to do testing that would otherwise need to be done manually. Once the testing has been automated, a large number of test cases can be validated quickly. This is most cost effective for software based products and devices that will have a long shelf life, because even minor patches over the lifetime of the application can cause features to break which were working at an earlier point in time. There are two ways to design the tests:

- Black box testing. The test developer has no knowledge of the inner workings of the program. The tests cover all the cases that an end user would run into. The completeness of the tests depends on the test developer's expertise using the application.

- White box testing. The test developer has full knowledge of the inner workings of the software based device. The tests ensure each pathway through the source code has been exercised and is working properly. Its completeness can be measured by code coverage metrics.

There are two general approaches to test automation:

- Graphical user interface testing. A testing framework generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behavior of the device is correct. The correctness of the device response as well as the timing of new test is controlled by the operator or person who is doing the testing.

- Test vector testing. Commonly used in the integrated circuit design industry, the test vector consists of a series of stimuli and expected device responses. Each test case is called a vector because it directs the device under test to a specific observable action. Elements types in the test vectors include logic levels (True and False), integer and floating point data types. A timing control is also required for automated testing that determines how long to wait for a device output to be asserted before declaring a fault and the time before moving on to the next test vector.

Test automation can be expensive, and it is usually employed in combination with manual testing. It can be made cost-effective in the longer term though, especially in regression testing. Regression testing is any type of software testing which seeks to uncover software regressions. Such regressions occur whenever software functionality that was previously working correctly, stops working as intended. Typically, regressions occur as an unintended consequence of program changes. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged.

Traffic Controller Test Automation

The automated tester, developed as part of this project, uses a graphical interface for manual testing. The scope of the traffic controller testing is those operations that can be generated by placing calls from vehicle detectors, pedestrian buttons, and preemption inputs. Only the traffic controller operations that are observable by asserting the signals of the NEMA TS1 A, B and C connections can be verified.

The automated tester can be operated in a static mode using the graphical user interface. The timing of input changes is strictly controlled by the person operating the testing system. Validation of traffic controller responses is the responsibility of the person testing the controller.

The automated testing uses XML script files to specify which inputs are activated, the timing of those activations and verifying the controller response(s). The advantage of automated testing is that tests can be repeated exactly and results are recorded automatically. Automated test can be repeated for long periods of time while subjecting the traffic controller to varying environmental conditions.

The testing software provided is for a limited set of NEMA TS1 controllers running a specific firmware version as specified by the Idaho Transportation Department. Other traffic controllers can be tested provided that the testing program has been modified to communicate with that specific controller. Software modifications to the testing software are required because there is no standard communications protocol used by various traffic controller manufacturers that allow the traffic controller settings to be read from the controller.

A version of the automated tester has been developed and tested that interfaces with NEMA TS2 type 1 and type 2 controllers. This version uses the National Transportation Communications for ITS Protocol (NTCIP) via either the asynchronous RS232 serial port or the Ethernet port. However, our investigation has shown that various vendors have significant differences in the interpretation of the NTCIP standard, and the automated testing software must still be verified for specific traffic devices running specific firmware versions.

Automated Tester: Hardware Requirements

Figure 5 illustrates the three hardware components requirements for the automated traffic controller testing system developed at the University of Idaho. All three devices must be powered separately with a 120VAC source. A PC with Windows XP or Vista operating system must have the proprietary software installed in accordance with guidelines in the user’s manual. The PC interface with a NIATT Controller Interface Device (CID) using a USB cable. The CID interfaces with a traffic controller using the A, B and C NEMA TS1 connectors.

Figure 5: Hardware Configuration for UI Automated Traffic Controller Testing.

Automated Tester: Software Requirements

The UI Automated Traffic Controller Tester software may be acquired by downloading the compressed file from the internet or from a memory storage media such as a CD or flash drive. The PC must have .NET Framework version 3.0 or higher installed on the PC. If .NET Framework version is not installed or the incorrect version is installed, the PC must be connected to the internet and the installation program will automatically download the required software provided the user has the appropriate computer access privileges.

Automated Tester: Components and Data Flow

The Automated Tester program is an interface to several components that when coupled together, provide the functions required. The key components are

CIDControl – Provides an interface with the CID API
TrafficControllers – Provides the framework to describe a traffic controller
Communications – Provides the framework to communicate with a traffic controller over serial or Ethernet

The CIDControl component provides all of the necessary functions to manage one or many CIDs. The CIDControl also provides events to which the Automated Tester can hook. These events are the basis for testing, observing, and conflict monitoring. Utilizing the Microsoft .NET Framework's event structure allows the Automated Tester to continuously monitor the CID (without taking all of the computer's resources). The Traffic Controllers component provides the types by which to classify a traffic controller. It is in this component where information extracted from the controller is formed into information to be used for testing. The Communications component is the basis for all intelligent communication between the PC and a traffic controller. The component implements the NTCIP protocols over both serial and Ethernet connections. The flow chart and data polling flow for the automated tester are presented in Figures 6 and 7.

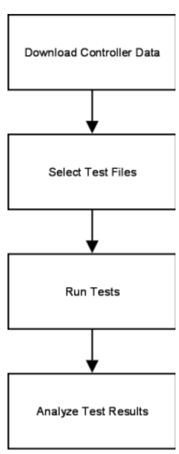


Figure 6: Automated Tester Flow Diagram.

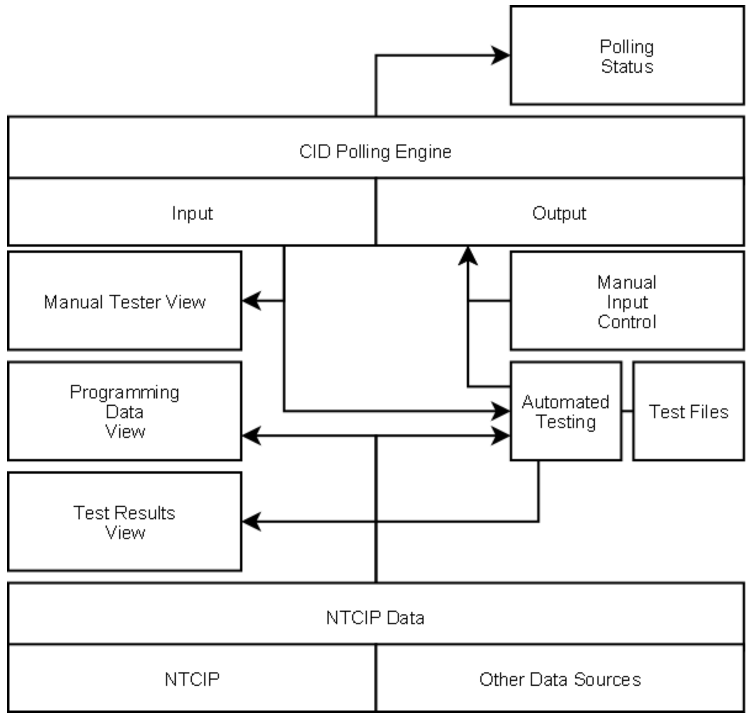


Figure 7: Automated Tester Data Polling Flow.

Automated Tester: Predefined Tests

Several important functions are covered in different tests presented here. A total of 14 tests were developed to test controller's ring parameters, phase parameters, and preempt parameters. Table 1 shows the tests and their corresponding functions.

Automated Tester: User's Manual

The automated tester user's manual is a web based publication that consists of seven chapters. Chapters 1 and 2 describe the automated tester installation and configuration. Chapter 3 shows how to add a controller. Chapters 4 through 6 describe test files management, running a test, and writing a test file, respectively. Chapter 7 provides reference for different test files. The contents of the user's manual can be viewed using a web browser such as Microsoft Internet Explorer (IE) or Mozilla Firefox. The automated tester user's manual contains instructions for installing the code on the PC. All test vector files are written using a text based XML (Extensible Markup Language.) Although Chapter 6 contains sufficient information necessary for developing new test vector files, tutorials for developing XML programs is available on the internet .[9]

Table 1: Controller Functions Covered in Different Tests

Functions		**Test Number**													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Ring Parameters	Phase In Use/Enable									x					
	Phase Sequence										x				
	Phase Omit Control						x								
Phase Parameters	Phase Concurrency														
	Phase Min Green	x	x	X											
	Phase Passage		x												
	Phase Max Green				x										
	Phase Yellow & Red			X											
	Min Recall (Veh Call)	x	x	X					x						
	Max Recall				x			x							
	Max Green 2							x							
	Red Rest								x						
	Dynamic Maxgreen					X									
	Dynamic Step					X									
	Phase Walk											x			
	Phase Ped Clearance											x			
	Ped Recall											x			
	Phase Time to Reduce														x
	Phase Time Before														x
	Phase Minimum Gap														x
Preempt Parameters	Preempt Minimum													x	
	Preempt Track Green												x		
	Preempt Track Phase												x		
	Preempt Dwell Phase												x		
	Preempt Delay													x	

References

(1) **Meyer, Mohaddes Associates**. *State of Idaho Intelligent Transportation Systems Strategic Plan.* 2000.

(2) **Li, Z., M. Kyte, and B. Johnson.** "A Controller Interface Device Based Suitcase Tester." pp 3107-3111 In *IECON '02: The 28th Annual Conference of the IEEE Industrial Electronics Society,*. 2002.

(3) **Siemens Intelligent Transportation Systems.** *NEMA TS-2 Controller Tester*. 2007. http://www.itssiemens.com/en/u_nav2162.html/, Accessed 6/26/2008.

(4) **NAZTEC Inc.** 2007. http://www.naztec.com/pdfs/testbox.pdf/ Accessed 6/26/2008.

(5) **Li, Z., A. Abdel-Rahim, B. Johnson, and M. Kyte**. "Design of Traffic Controller Automated Testing Tool". *Transportation Research: Part C: Emerging Technologies*, Vol. 16C, No. 3, (June 2008): 277-293.

(6) **National Electrical Manufacturers Association.** *Traffic Control Systems. NEMA Standards.* Publication No. TS 1-1989. Washington, DC: National Electrical Manufacturers Association. 1989.

(7) **National Electrical Manufacturers Association.** *Traffic Controller Assemblies. NEMA Standards,* Publication No. TS 2-1992. Washington, DC: National Electrical Manufacturers Association, 1992.

(8) **California Department of Transportation (Caltrans).** *Transportation Electrical Equipment Specifications (TEES)*. Sacramento, CA: California Department of Transportation, March 1997.

(9) XML Tutorial 2008. http://www.w3schools.com/xml/default.asp Accessed 9/11/2008.