

FINAL REPORT

To

The Florida Department of Transportation
Research Office

On Project

"Development of Automated Testing Tools for Traffic
Control Signals and Devices"

FDOT Contract Number BDK83-977-08

June 30, 2012

By

Leonard J. Tung
Department of Electrical and Computer Engineering
FAMU-FSU College of Engineering, Florida State University

DISCLAIMER

The opinions, findings, and conclusions expressed in this publication are those of the authors and not necessarily those of the State of Florida Department of Transportation.

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle Development of Automated Testing Tools for Traffic Control Signals and Devices		5. Report Date June 30, 2012	
		6. Performing Organization Code	
7. Author(s) Leonard J. Tung		8. Performing Organization Report No.	
9. Performing Organization Name and Address Florida State University Tallahassee, FL 32306		10. Work Unit No. (TRAVIS)	
		11. Contract or Grant No. BDK83-977-08	
12. Sponsoring Agency Name and Address Florida Department of Transportation 605 Suwannee St. MS 30 Tallahassee, Florida 32399		13. Type of Report and Period Covered Final Report	
		14. Sponsoring Agency Code	
15. Supplementary Notes Prepared in cooperation with the USDOT and FHWA			
16. Abstract Through a coordinated effort among the electrical engineering research team of the Florida State University (FSU) and key Florida Department of Transportation (FDOT) personnel, an automated testing system for National Electrical Manufacturers Association (NEMA) TS2 Type-1 Actuated Signal Controller (ASC) has been developed and constructed. The system developed consists of the following: <div style="margin-left: 40px;"> <p>A laptop with proper ports and software,</p> <p>A Personal Computer Memory Card International Association (PCMCIA) card by Quatech,</p> <p>A device for the interface between an ASC and the Quatech card,</p> <p>A total of 20 automated testing programs covering all the functionalities of an ASC,</p> <p>An executable C# Windows Console application to execute all the automated testing programs: <i>ASCAutoTester.exe</i>,</p> <p>A user manual for the automated ASC testing system, and</p> <p>A compact disk (CD) containing all program codes and documents of the project</p> </div>			
17. Key Word ASC, Automated Testing Programs, Python scripting language, Alternative NTCIP Testing Software (ANTS)		18. Distribution Statement No Restriction	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 48	22. Price

ACKNOWLEDGEMENTS

The authors would like to express their sincere appreciation to Jeffrey M. Morgan, Carl A. Morse, and Derek J. Vollmer of the Florida Department of Transportation for the guidance and support that they provided on this project.

EXECUTIVE SUMMARY

The lack of an adequate testing system has hampered the efforts of key personnel at the Florida Department of Transportation (FDOT) to conduct comprehensive testing of Actuated Signal Controllers (ASC) submitted by various manufacturers/vendors for certification. The objective of the project is to design and implement an automated testing system for efficient, thorough, and accurate testing of ASCs.

Through a coordinated effort among the electrical engineering research team of the Florida State University (FSU) and key Florida Department of Transportation (FDOT) personnel, an automated testing system for National Electrical Manufacturers Association (NEMA) TS2 Type-1 ASC has been developed and constructed.

The system developed consists of the following:

- A laptop with proper ports and software,
- A Personal Computer Memory Card International Association (PCMCIA) card by Quatech,
- A device for the interface between an ASC and the Quatech card,
- A total of 20 automated testing programs covering all the functionalities of an ASC,
- An executable C# Windows Console application to execute all the automated testing programs: *ASCAutoTester.exe*,
- A user manual for the automated ASC testing system, and
- A compact disk (CD) containing all program codes and documents of the project

The automated testing system is somewhat manufacturer dependent due to the discrepancies among the manufacturers in their interpretations of many of the ASC specifications. The development and the eventual adoption of National Transportation Communications for ITS (Intelligent Transportation Systems) Protocol (NTCIP) have demonstrated the possibility of an NTCIP based autonomous testing system that is manufacturer independent. Additional work is needed to develop a manufacturer independent autonomous testing system that combines the Automated Testing System developed in this project and previous NTCIP test procedures developed at FDOT Traffic Engineering Research Lab (TERL).

This report contains an evaluation of the McCain-NIATT Controller Interface Device (CID II) Hardware in the Loop research project final report and the Automated ASC Testing Device User Manual developed during the current research project. Other material developed during the current research project such as project administration files and ASC automated software can be found on the accompanying CD (in HTML web format) or via the projects web site at: <http://eng.fsu.edu/~tung/terl/index.htm>

TABLE OF CONTENTS

I. Introduction	1
I.1. Background	1
I.2. Research Objectives and Supporting Tasks	1
II. Literature Review	3
III. Areas of Work and Scope	4
IV. Results and Products	6
V. Conclusion	8
Appendix A	10
Appendix B	15

LIST OF TABLES

Table 1: Areas of Work and Scope	4
Table 2: Results and Products	6

LIST OF FIGURES

Figure 1: Quatech PCMCIA Card Drivers CD	17
Figure 2: Econolite Controller	18
Figure 3: Front View of the Automated Testing Interface Unit	18
Figure 4: Rear View of the Automated Testing Interface Unit	19
Figure 5: Quatech PCMCIA SDLC Card	19
Figure 6: Screenshot of Windows Command Prompt to Begin the Test	20
Figure 7: Set-Up of the Testing Equipment with an ASC	20
Figure 8: Screenshot of the Subdirectory ASC_Automated_Tests.....	21
Figure 9: Screenshot Listing the Various Tests	22
Figure 10: Screenshot indicating the status of “Program Is Now Running”	22
Figure 11: Screenshot Indicating the Completion of the Chosen Test	23
Figure 12: Screenshot of a Report Generated by the Minimum Green Phase Timing Test ...	24
Figure 13: Screenshot of a Report Generated by the Minimum Green Phase Timing Test with Fails.....	25
Figure 14: Screenshot of a Single Min/Max Phase Timing Test.....	25
Figure 15: Screenshot of a Report Generated by the Minimum Red Phase Timing Test.....	26
Figure 16: Screenshot of the Vehicle Call Coordination Test	26
Figure 17: Screenshot of a Report Generated by the Vehicle Call Coordination Test.....	27
Figure 18: Screenshot of the Pedestrian Call Coordination Test.....	27
Figure 19: Screenshot of a Report Generated by the Pedestrian Call Coordination Test.....	28
Figure 20: Screenshot of the Preemption Call Coordination Test	28
Figure 21: Screenshot of a Report Generated by the Preemption Call Coordination Test.....	29
Figure 22: Screenshot of the Preemption Priority Test.....	29
Figure 23: Screenshot of a Report Generated by the Preemption Call Coordination Test.....	30
Figure 24: Screenshot of the ASC Flash Test.....	30
Figure 25: Screenshot of a Report Generated by the ASC Flash Test.....	31
Figure 26: Screenshot of the Overlap Configuration Test	31
Figure 27: Screenshot of a Report Generated by the Overlap Configuration Test.....	32
Figure 28: Screenshot of the Channel Mapping Test.....	32
Figure 29: Screenshot of a Report Generated by the Channel Mapping Test	33
Figure 30: Screenshot of the Flashing Yellow Test.....	33
Figure 31: Screenshot of a Report Generated by the Flashing Yellow Test.....	34
Figure 32: Screenshot of the Stop Time Test.....	34
Figure 33: Screenshot of a Report Generated by the Stop Time Test	35
Figure 34: Screenshot of the ASC Phase Startup Test.....	35
Figure 35: Screenshot of a Report Generated by the ASC Phase Startup Test.....	35
Figure 36: Screenshot of the Coordination Test	36
Figure 37: Screenshot of a Report Generated by the Coordination Test	36

LIST OF ABBREVIATIONS

Abbreviation	Full Description
AC	Alternating Current
ANTS	Alternative NTCIP Testing Software
APL	Approved Product List
ASC	Actuated Signal Controller
ATIU	Automated Testing Interface Unit
CD	Compact Disc
CID	Controller Interface Device
CORSIM	Corridor Simulation
COTS	Commercial-off-the-shelf
DC	Direct Current
EW	East-West bound
FDOT	Florida Department of Transportation
FSU	Florida State University
HILSCI	Hardware in the Loop (Simulator) Component Integrator
ITS	Intelligent Transportation Systems
MMU	Malfunction Management Unit
NEMA	National Electrical Manufacturers Association
NIATT	National Institute for Advanced Transportation Technology
NS	North-South bound
NTCIP	National Transportation Communications for ITS Protocol
PC	Personal Computer
PCMCIA	Personal Computer Memory Card International Association
PED	Pedestrian
PI	Principal Investigator
RTF	Rich Text Format
SDLC	Synchronous Data Link Control protocol
TERL	Traffic Engineering Research Lab
TRF	Train Simulator Traffic Pattern Format
TSIS	Traffic Software Integrated System

I. Introduction

I.1. Background

The Florida Department of Transportation (FDOT) is required by State law (Section 316.0745, F.S.) to develop and publish specifications for traffic control signals and traffic control devices and certify devices that meet these specifications before they are allowed to be sold or purchased as part of a traffic control system within the state of Florida. To fulfill this requirement, the FDOT Traffic Engineering and Operations Office, Traffic Engineering Research Lab (TERL) develops specifications and performs operational testing to certify devices submitted by product manufacturers that are in compliance with published FDOT specifications.

One common traffic control device submitted by manufacturers that requires testing is the Actuated Signal Controller (ASC), referred to hereafter as “traffic signal controller.” The traffic signal controller is basically a microcomputer located at the intersection that processes various inputs and triggers outputs that control traffic signals, pedestrian signals, and other electronic devices that comprise a signalized intersection.

Because the traffic signal controller is designed to accommodate a wide variety of intersection types, ranging from simple to extremely complex, the configurable nature of the traffic signal controller makes testing the device a particularly challenging and time consuming exercise. Comprehensive testing of the traffic signal controller requires the device to be manually operated and monitored in a consistent and controlled fashion over a long period of time. Recent National Transportation Communications for ITS (Intelligent Transportation Systems) Protocol (NTCIP) requirements have only increased the complexity of testing these devices. The TERL is able to perform reasonably effective cursory reviews of fundamental traffic signal controller features; however, it is not practical to conduct more thorough and comprehensive testing manually on each traffic signal controller received for evaluation due to the intensive and time consuming nature of these tests.

A handful of commercial testing tools are marketed for these purposes, but first-hand trials of these products by the TERL have uncovered a number of shortcomings that prevent them from being viable for the purposes of certification testing.

Due to the above limitations, testing of the traffic signal controller is being performed in a labor-intensive, somewhat ad-hoc manner. This research activity is being proposed to overcome these limitations by developing a set of automated test procedures and test tools that would yield efficient, consistent and effective results in a timely manner.

I.2. Research Objectives and Supporting Tasks

The main objective of this project is to produce a set of procedures and tools that can be used to perform automated testing of ASC against FDOT technical requirements. The

pass/fail results of testing each requirement will be the basis for accepting or rejecting products submitted for listing on the FDOT Approved Product List (APL).

To achieve this objective, the following tasks are anticipated:

- Research and review of past efforts applicable to this project.
- Research, review, and selection of existing Commercial-off-the-shelf (COTS) products required for this project.
- System requirements development.
- System design and design reviews.
- System implementation.
- System testing and validation.
- Implementation of production test environment.
- Documentation.
- Training.

II. Literature Review

Among the effort in the development of automated testing tools for ASC, the most cited publication is the work done at National Institute for Advanced Transportation Technology (NIATT). The automated testing tool developed by Zhen Li et al at NIAFF [1], would make it possible for the design and implementation of an automated testing system for ASC. The tool developed by Li et al requires a controller interface device (CID) installed between a host testing computer and an ASC. After the evaluation of the NIATT CID (CID II) was completed, it was determined that such a CID without the capability of supporting SDLC (Synchronous Data Link Control) protocol would not be adequate for the development of a fully automated testing system capable of a comprehensive testing of ASC (Appendix A). To ensure the success of the design and implementation of such an automated testing system, researchers decided that effort would be focused on NEMA TS2 Type-1 ASC with NTCIP requirements [2].

III. Areas of Work and Scope

The key areas of work alongside their scopes are listed in Table 1.

Table 1: Areas of Work and Scope

Area of Work	Scope
<i>Research and review of past efforts applicable to this project.</i>	These tasks are expected to include, but not be limited to, gaining familiarity with Florida's existing specification for traffic signal controllers, the configuration and operation of traffic signal controllers, the NTCIP standards related to this effort (i.e.: NTCIP 1202, 1201, 8007, etc.), the current tools and development environments used within the TERL, and other background information gathering that may be necessary.
<i>Research, review, and selection of existing Commercial-off-the-shelf (COTS) products required for this project.</i>	These tasks are expected to include, but not be limited to, investigating, identifying, and procuring hardware and software to support the project work. This project will require some computer hardware and software. Computer hardware and software should conform to current FDOT standards insofar as practical.
<i>System requirements development.</i>	The Principal Investigator (PI) and his staff will be responsible for identifying FDOT user needs and developing a detailed requirements document that captures the expected operational functionality of the automated testing system in accordance with system engineering best practices. These requirements will then guide the design and implementation of the automated testing system.
<i>System design and design reviews.</i>	The PI and his staff will be responsible for creating flow charts, pseudo code, and other documentation detailing the high and low level concepts of the system. The team will be responsible for modifying these designs as the project progresses to reflect the actual operation of the system.
<i>System implementation.</i>	The PI and his staff will be responsible for implementing the system as outlined in the design phase. This will include, but not be limited to, producing code to interface the test software with the Hardware in the Loop device, modify existing scripts currently written for the test software to utilize the interface with the Hardware in the Loop device, and create new test procedures and scripts to thoroughly test the functionality of the traffic signal controller. All test procedures created shall follow the format outlined in NTCIP 8007.
<i>System testing and validation.</i>	The PI and his staff will be responsible for thoroughly testing the system in order to identify any design flaws or bugs within the system. If design flaws are found, the team shall modify the design, implement the design changes, and retest the system. If bugs are discovered, the team shall isolate and correct the bugs and retest the system to verify proper operation.

<p><i>Implementation of production test environment.</i></p>	<p>The PI and his staff will be responsible for packaging the test software into an installer that can be easily distributed. The PI and his staff will demonstrate to FDOT that the installer can be used to easily create a production test environment on a “clean” PC (Personal Computer) target (PC should only possess a clean installation of Windows XP Professional).</p>
<p><i>Documentation.</i></p>	<p>The PI and his staff will be responsible for creating support and design documentation. This will include, but is not limited to, a user manual for the final packaged system (including software installation, operation, and hardware setup), flow charts detailing software module interactions and software design concepts, and comments within the source code to make it easy for someone other than the programmer to understand how the code works.</p>
<p><i>Training.</i></p>	<p>At no additional cost to FDOT, the PI and his staff will be responsible for providing up to 80 hours training to various FDOT employees on how to install, configure, and operate the final system, as well as comprehend the output from the various tests developed. Training will be conducted at the FDOT-TERL in Tallahassee, Florida.</p>

IV. Results and Products

All the results and products of this research project are compiled and stored in the accompanying compact disc (CD). The summary of results and products is presented in Table 2.

Table 2: Results and Products

Area of Work	Results and Products
<i>Research and review of past efforts applicable to this project.</i>	Review of the paper titled <i>Design of Traffic Controller Automated Testing Tool</i> , by Zhen Li, Ahmed Abdel-Rahim, Brian Johnson, and Michael Kyte, published by Elsevier, 2008 [1].
<i>Research, review, and selection of existing Commercial-off-the-shelf (COTS) products required for this project.</i>	Report titled <i>Evaluation of the McCain-NIATT Controller Interface Device (CID II) "Hardware in the Loop"</i> (Appendix A).
<i>System requirements development.</i>	Testing software based on the functionalities of the NEMA TS2 Type-2 ASC [2].
<i>System design and design reviews.</i>	A Laptop with proper ports and software. A PCMCIA card by Quatech. A device for the interface between an ASC and the Quatech card.
<i>System implementation.</i>	A total of 20 automated testing programs covering all the functionalities of the NEMA TS2 Type-2 ASC (in the accompanying CD).
<i>System testing and validation.</i>	Sample testing reports for prevalent traffic scenarios at an intersection with an ASC.
<i>Implementation of production test environment.</i>	An executable C# Windows Console application to execute all the automated testing programs: <i>ASCAutoTester.exe</i> (in the accompanying CD).
<i>Documentation.</i>	Automated ASC Testing Device User Manual (Appendix B). Final report on CD in Web format.
<i>Training.</i>	Initially scheduled and started on January 19, 2012 for twice a week with one hour for each training session. Switched to three times a week with one hour each from February 3 through February 24, 2012.

The PCMCIA card by Quatech usually offers about 3 minutes of stable testing environment till some synchronization problem causes the testing program to freeze. Once the testing program freezes, it is necessary and time-consuming to reset the entire system. A 3-minute run time is proven to be restrictive to 3 of the 20 testing programs:

Maximum Green Phase Timing Test

Maximum Yellow Phase Timing Test

Maximum Red Phase Timing Test

For these Maximum Phase Timing Tests, only one of the 8 phases can be set at the maximum phase time while the rest of the phases are set at phase times shorter than the maximum.

The manufacturer of the PCMCIA SDLC Card, Quatech, was contacted and had provided assistance. However, they were unable to resolve the problem.

To overcome this limitation, two Senior Design Projects (with 3 undergraduates each) have been commissioned in association with the Department of Electrical and Computer Engineering at FSU to develop a system to replace the PCMCIA card by Quatech and the interface between the card and an ASC for a more stable testing environment.

V. Conclusion

The FSU electrical engineering research team and key FDOT personnel have developed an Automated Testing System for NEMA TS2 Type-2 ASCs. Unfortunately, ASC manufacturers typically use proprietary software and hardware in the design of ASC. There are discrepancies among the manufacturers in their interpretations of many of the ASC specifications. Consequently, the Automated Testing Tools developed by researchers so far are manufacturer dependent. The development and the eventual adoption of NTCIP have demonstrated the possibility of an NTCIP based autonomous testing system which is manufacturer independent. Additional work is needed to develop a manufacturer independent autonomous testing system that combines the Automated Testing System developed in this project and previous NTCIP test procedures developed at FDOT Traffic Engineering Research Lab (TERL).

References

1. Zhen Li, Ahmed Abdel-Rahim, Brian Johnson, and Michael Kyte, "Design of traffic controller automated testing tool," *Transportation Research Part C* 16 (2008) 277–293, Elsevier
2. NEMA Standards Publication TS2-2003 v02.06, *Traffic Controller Assemblies with NTCIP Requirements*.

Appendix A

Evaluation of the McCain-NIATT Controller Interface Device (CID II) “Hardware in the Loop”

By

Tim Walton
Department of Electrical and Computer Engineering
Florida State University

Table of Contents

- I. Purpose of Evaluating
- II. Pros and Cons from Evaluation
 - 1. Pros
 - 2. Cons
- III. Questions
- IV. FDOT-TERL Conclusion

Product Evaluation Activity Report

Activity Period: December 10, 2009 – December 15, 2009

Subject: Evaluation of the McCain-NIAT Controller Interface Device (CID II) “Hardware in the Loop”

FDOT-TERL: Jeff Morgan, (FDOT Project Manager)
Dr. Leonard Tung, (FSU Principal Investigator)
Tim Walton, (FSU Research Associate)

CID Contacts: McCain Traffic Supply, Inc.
NIATT
University of Idaho
Ken Courage (University of Florida Professor)

I. Purpose of Evaluating

The McCain-NIATT Controller Interface Device (CID), also referenced as “Hardware in the Loop,” can be very useful in the current research project, “Development of Automated Testing Tools for Traffic Control Signals and Devices.” Researchers believe that the device may increase the speed of the currently tedious and rigorous testing process. The potential of the CID device is not fully known. Researchers are studying its potential to develop an automated process that tests signal controllers. Ideally, researchers will be able to obtain access to the programming code of this CID device and expand upon its functions to a level that makes possible the development of an automated testing system for ASC.

II. Pros and Cons from Evaluation

1. Pros

- (a) Scripting by intervals and signals is made possible
- (b) Timing of traffic signals can be adjusted
- (c) Pedestrian phase signals can be made to an extent
- (d) Intervals can be advanced
- (e) Intersection layout is adjustable up to 4 lanes EW (East-West *bound*) and 3 lanes NS (North-South *bound*)
- (f) Changeable cycles for varied scripted testing
- (g) Presence of a logging and testing tool

2. Cons

- (a) Pedestrian phases are either always high, or always low
- (b) No true preemption, just timed interval advances
- (c) No set proper output signals for logging and comparison mode
- (d) Cannot program/code scripting, just built in function calls
- (e) Intersection layout is only up to 8 phases with no apparent right turn leads
- (f) No manual control settings such as FLASH mode
- (g) No confliction allowed to test proper outputs such as a Flashing Test
- (h) Bulky and a computer is needed to operate
- (i) Not compatible with NEMA TS2 Type 1 signal controllers, only NEMA TS1 and NEMA TS2 Type 2 via A/B/C connectors.

III. Questions

1. Is CORSIM (Corridor Simulation) linking a possible solution to advanced testing procedures via HILSCI (Hardware in the Loop Simulator Component Integrator)?
2. Is the “right turn lead” phase possible to properly execute?
3. Is there a way to access/interface lower level programming of the CID device?

4. What is truly meant by forcing off a signal or cycle when executing a script?
5. What exactly are TSIS (Traffic Software Integrated System) and TRF (Train Simulator Traffic Pattern Format) files?
6. Is connection to a NEMA TS2 controller possible with modifications?

IV. FDOT-TERL Conclusion

It is apparent at this time that the McCain-NIATT Controller Interface Device (CID II), aka “Hardware in the Loop,” is going to produce less than desirable automated testing. Too many limitations are present currently. Further research can be done in regards to some final proposed questions, but currently this device has proven to be a dead end in aiding in the production of fully automated testing tools.

Appendix B

Automated ASC Testing Device User Manual

By

Tim Walton

Department of Electrical and Computer Engineering
Florida State University

I. Introduction

The user manual contains the following sections to help simplify the set-up and use of the automated testing device for the Actuated Signal Controller (ASC):

Software: explains the software installations necessary for proper use of the testing device.

Hardware: explains the interface for the connection to an ASC.

Automated Tests: walk-through of how to execute each automated testing script.

Addendum: description of what is going on for each test.

II. Software

The testing software requires the additional installations of the scripting language software Python, the editing software Notepad++, and the Quatech PCMCIA card drivers. The procedures are outlined in the following steps.

Step 1: Install the software Python 2.5 on the desired laptop to be used for ASC automated testing.

- Python 2.5 is a freeware and can be downloaded from user's choice of website.
- Once it is installed, be sure to copy and paste the folder named *ASC_Automated_Tests* which contains all automated testing scripts coded in Python and is separately supplied in a compact disk (CD) into the directory of your hard drive, say C:\Python25.

Step 2: Install the software Notepad++ on the desired laptop.

- Notepad++ is also a freeware that can be downloaded from user's choice of website.
- With Notepad++, all scripts coded in Python can be viewed and edited as needed. To edit any automated testing script, just right-click and choose "Edit with Notepad++" to view the code.

Step 3: Install the Quatech PCMCIA card drivers.

- The CD containing the drivers is pictured in Figure 1.

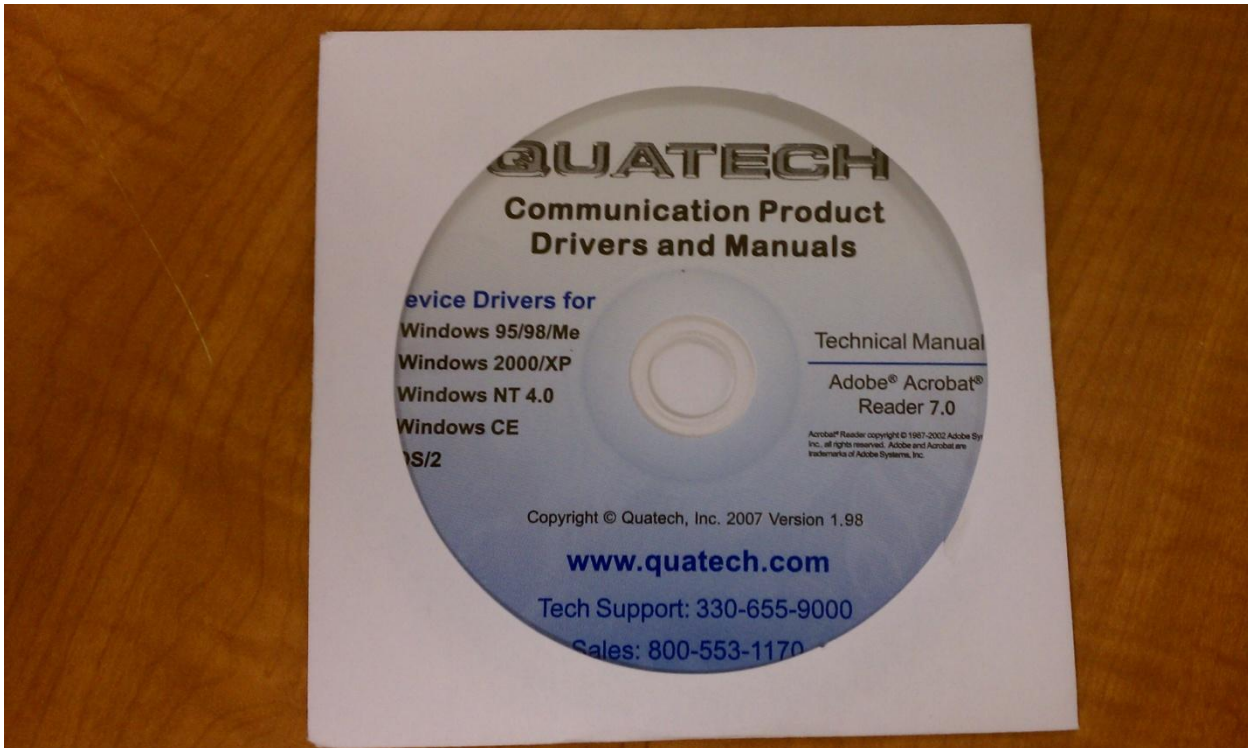


Figure 1: Quatech PCMCIA Card Drivers CD

- The most up-to-date driver that stops the freezing issues is the “QuaSYS” system file found in the folder named "Non-Freezing Driver" which is contained in the "Quatech Folder."
- Drop this into the folder on the machine being used for testing, C:\Windows\System32\Drivers.

III. Hardware

The testing hardware consists of the following (see Figure 7):

A laptop with proper ports and software,

A Personal Computer Memory Card International Association (PCMCIA) card by Quatech, and

A device known as the Automated Testing Interface Unit (ATIU) for the interface between an ASC and the Quatech card.

The installation procedures are outlined in the following steps.

Step 1: Acquire the ASC to be tested and an AC (Alternating Current) power cord for the ASC.

- Shown in Figure 2 is an Econolite traffic controller that was used during the system development.



Figure 2: Econolite Controller

Step 2: Locate the Automated Testing Interface Unit (ATIUI) for the SDLC (Synchronous Data Link Control) connection.

- Shown in Figure 3 is the front view of the ATIUI with a 25-pin port for the connection to a PCMCIA (Personal Computer Memory Card International Association) card.



Figure 3: Front View of the Automated Testing Interface Unit

- Plug in the gray SDLC cable to the 15-pin Port 1 Interface on the front of the ASC.
- Plug in a DC (Direct Current) source via banana connectors with a constant voltage of 5 Volts. The connectors are located at the rear of the ATIUI as shown in Figure 4. Set the toggle switch in its neutral position, in the middle.

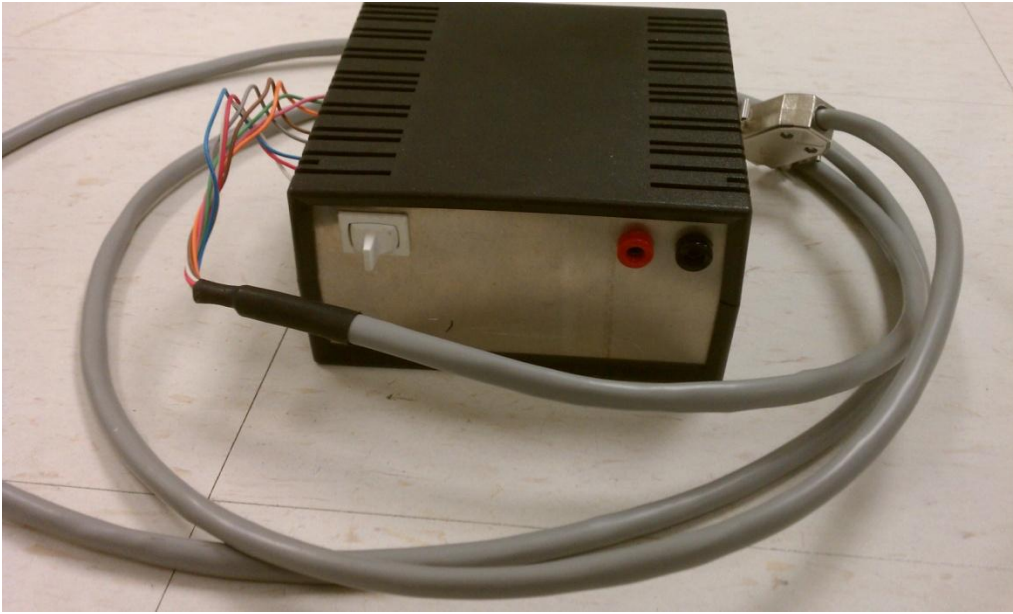


Figure 4: Rear View of the Automated Testing Interface Unit

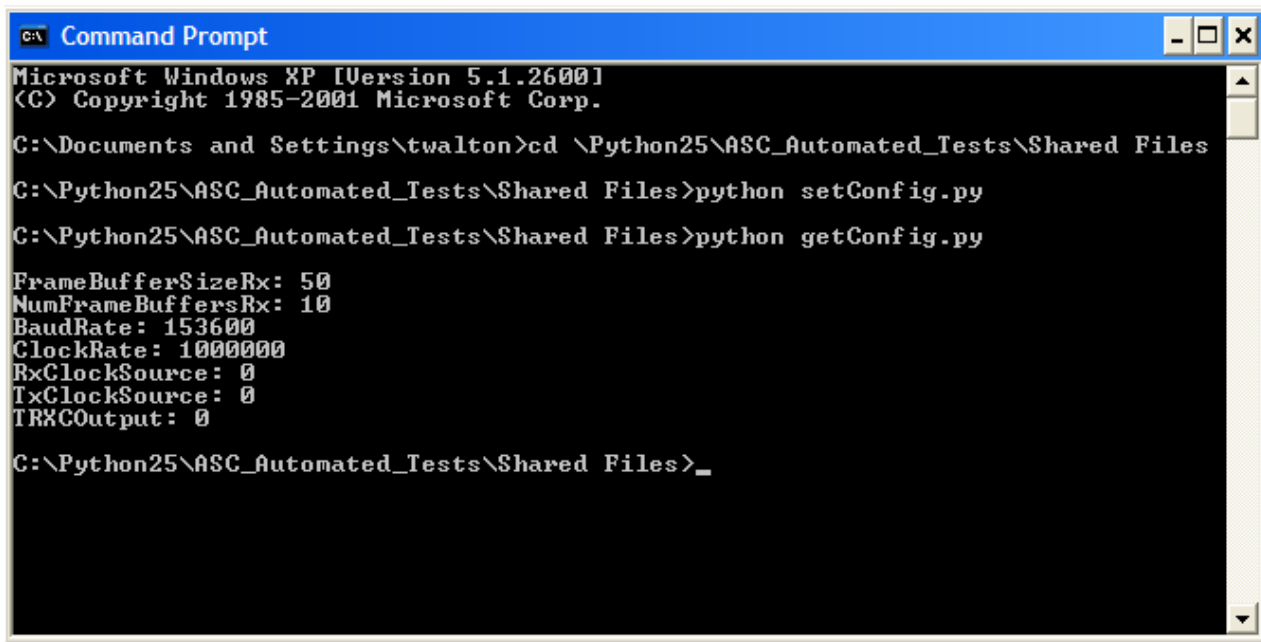
Step 3: Locate the Quatech PCMCIA SDLC card as shown in Figure 5.



Figure 5: Quatech PCMCIA SDLC Card

- Connect this card to the afore-mentioned 25-pin port in the front of ATIU and the laptop.
- Once this is recognized as a working device by the laptop, open up a Windows Command Prompt.
 - Firstly, type: “cd \Python25\ASC_Automated_Tests\Shared Files”
 - Then type: “python setConfig.py”
 - Finally, type: “python getConfig.py”
 - This will prompt some lines of text to be displayed. Sequentially, make sure the *FrameBufferSizeRx* field is set to 50, the *NumFrameBuffersRx* field is set to 10,

the *BaudRate* is set to 153600, and the *ClockRate* is set to 1000000. Shown in Figure 6 is the screenshot of a sample Console Window.



```
C:\> Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\twalton>cd \Python25\ASC_Automated_Tests\Shared Files
C:\Python25\ASC_Automated_Tests\Shared Files>python setConfig.py
C:\Python25\ASC_Automated_Tests\Shared Files>python getConfig.py

FrameBufferSizeRx: 50
NumFrameBuffersRx: 10
BaudRate: 153600
ClockRate: 1000000
RxClockSource: 0
TxClockSource: 0
TRXOutput: 0

C:\Python25\ASC_Automated_Tests\Shared Files>_
```

Figure 6: Screenshot of Windows Command Prompt to Begin the Test

- The final Set-Up should appear as in Figure 7. As shown in the figure, the ASC under test is in the back. In front of the ASC, from left to right, are a 5-volt DC power supply, the ATIU, and the laptop with the Quatech PCMCIA SDLC card.

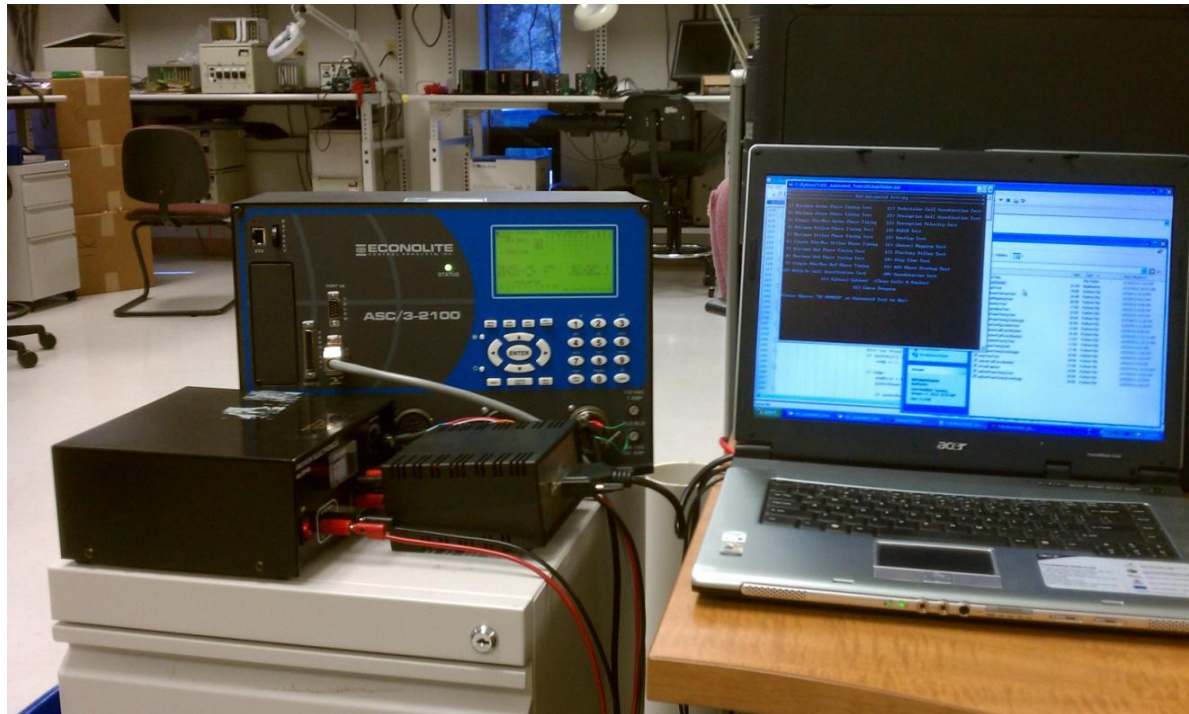


Figure 7: Set-Up of the Testing Equipment with an ASC

IV. Automated Tests

There are 20 automated tests developed for testing the functionalities of an ASC. All 20 tests are compiled into an executable file named "*ASCAutoTester.exe*". The testing procedures are outlined in the following steps.

Step 1: Open up the application: *ASCAutoTester.exe*

- This is found in the folder named *ASC_Automated_Tests* and is highlighted in the “Tile-View” of the folder as shown in Figure 8.

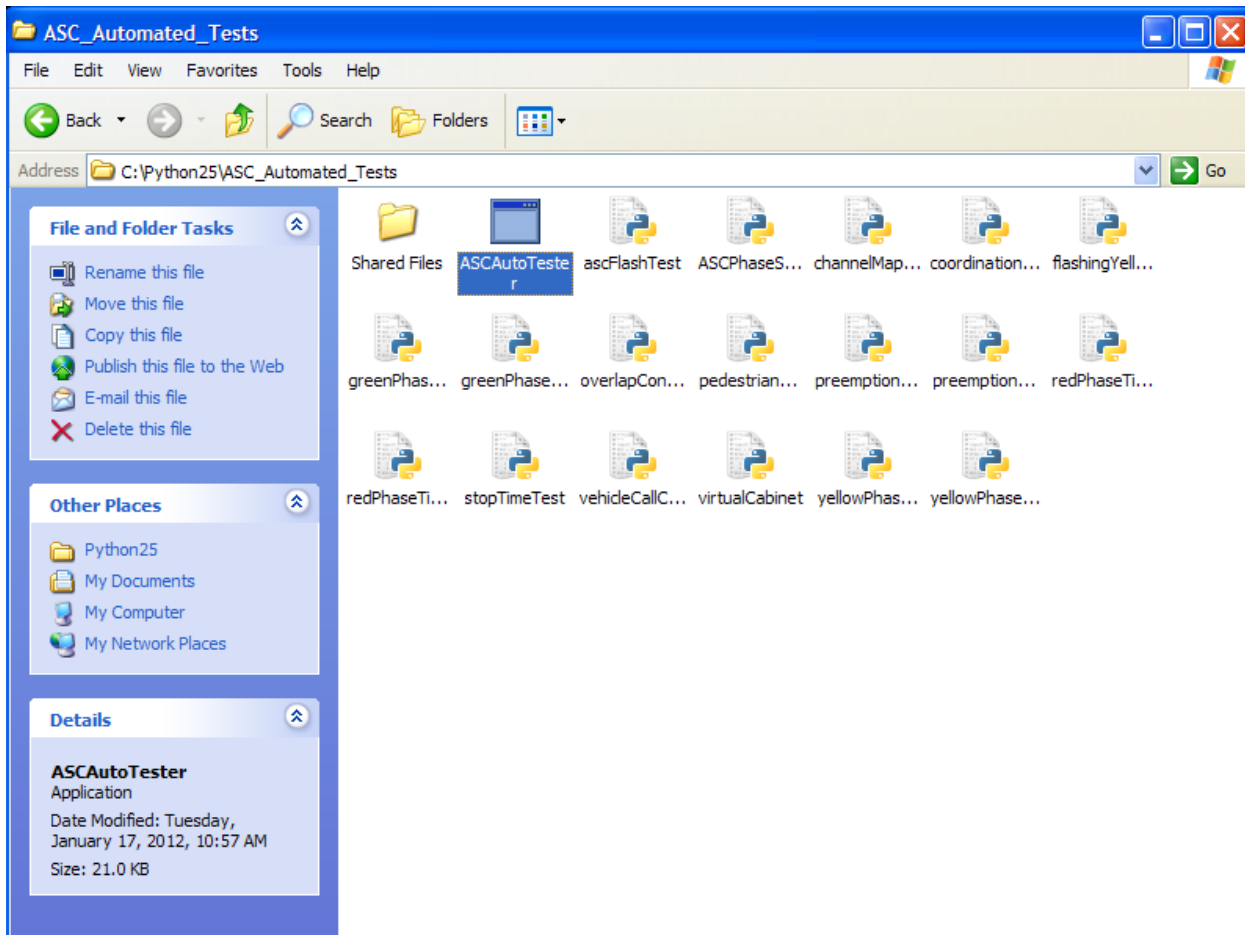


Figure 8: Screenshot of the Subdirectory *ASC_Automated_Tests*

- Make sure the ASC has the "TS2 Type Cabinet" communication method enabled.
- Also, make sure the "MMU TO CU SDLC EXTERNAL START" option is disabled under the settings in the ASC menu titled "SDLC PORT 1 CONFIGURATION."
- To verify functionality, choose the *Virtual Cabinet* option, Number: 21, as shown in Figure 9.

```

C:\Python25\ASC_Automated_Tests\ASCAutoTester.exe
----- ASC Automated Testing -----
1> Minimum Green Phase Timing Test      11> Pedestrian Call Coordination Test
2> Maximum Green Phase Timing Test      12> Preemption Call Coordination Test
3> Single Min/Max Green Phase Timing    13> Preemption Priority Test
4> Minimum Yellow Phase Timing Test     14> FLASH Test
5> Maximum Yellow Phase Timing Test     15> Overlap Test
6> Single Min/Max Yellow Phase Timing    16> Channel Mapping Test
7> Minimum Red Phase Timing Test        17> Flashing Yellow Test
8> Maximum Red Phase Timing Test        18> Stop Time Test
9> Single Min/Max Red Phase Timing      19> ASC Phase Startup Test
10> Vehicle Call Coordination Test      20> Coordination Test
      21> Virtual Cabinet <Clear Calls & Faults>
      22> Close Program

Please Choose 'BY NUMBER' an Automated Test to Run: _

```

Figure 9: Screenshot Listing the Various Tests

- Choose 40 so the "Error and Fault-Free Virtual Cabinet" environment will run for forty seconds to verify the "Set-Up" is correct for proper automated testing. If all the minimum phase times are set, this allotment will clear any and all internal calls that are placed at start-up due to this environment running a "Red Rest" state on both rings.
- The console will display "Program Is Now Running" while the "Virtual Cabinet" environment is executing and then will notify when the program has ended as shown in Figure 10.

```

Please Choose 'BY NUMBER' an Automated Test to Run: 21
For the Virtual Cabinet, One Parameter needs to be defined:
Desired Duration <in Seconds> for Virtual Cabinet to Run: 40
Program is Now Running
Virtual Cabinet Environment Has Ended.

```

Figure 10: Screenshot indicating the status of "Program Is Now Running"

- After this, the original header with all the tests listed as options will be displayed again.

Step 2: Configure the ASC to run the first nine automated tests. (Phase Timing Tests)

- Configure the ASC to be tested into a "Single-Ring Setup" with "No Phase Compatibilities."
- Make sure the "SDLC Options for Port 1 Configurations" has "Terminal Facility BIU 1," "Terminal Facility BIU 2," "Terminal Facility BIU 3," and "TS2/MMU TYPE CABINET" enabled. The rest are "No" and/or "Disabled" for now.
- Next, assign the appropriate amount of times "Minimum" or "Maximum" for the signal head states of the phases to be tested (times for the red, the yellow, & the green signals).

- Finally, set "Internal ASC Phase Recalls" active on all eight phases.

Step 3: Executing the Phase Timing Tests.

- All of the minimum and maximum timing tests are run in similar ways. Shown in Figure 11, as an example, is Test #1 for testing "Minimum Green Times" for all eight phases.

```

Please Choose 'BY NUMBER' an Automated Test to Run: 1
For the Minimum Green Phase Timing Test, Parameters need to be defined:
Phase 1 Minimum Green Time: 1
Phase 2 Minimum Green Time: 1
Phase 3 Minimum Green Time: 1
Phase 4 Minimum Green Time: 1
Phase 5 Minimum Green Time: 1
Phase 6 Minimum Green Time: 1
Phase 7 Minimum Green Time: 1
Phase 8 Minimum Green Time: 1
Test is Now Running
Test Complete. Reference Report in Folder: 'ASC_Automated_Tests' for Results.

```

Figure 11: Screenshot Indicating the Completion of the Chosen Test

- After "1" is typed in, Test #1 would prompt for parameters needed to run the test.
- For each of the specified phases, the defined allowable minimum green time of 1 second is inputted.
- The test then runs and prompts upon completion to find the report in the same folder as all the tests and current console application be used; "ASC_Automated_Tests".
- The report will generate in the form of date/time followed by the test type: "2012_01_31_13_49_32_MinimumGreen_TestReport"
- Shown in Figure 12 is an example of a successful execution.

Minimum Green Phase Timing Test

Phases	Resulting Test Information	Pass / Fail
1	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass
2	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass
3	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass
4	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass
5	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass
6	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass
7	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass
8	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass

Figure 12: Screenshot of a Report Generated by the Minimum Green Phase Timing Test

- Whenever errors are present, the row (or phase) in issue is shown in bold. Shown in Figure 13, as an example, is how the test report appears when the min times for the even phases are set to be 5 seconds in the timing plans of the ASC rather than the 1 second.

Minimum Green Phase Timing Test

Phases	Resulting Test Information	Pass / Fail
1	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass
2	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 5.0 seconds	Fail
3	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass
4	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 5.0 seconds	Fail
5	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass
6	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 5.0 seconds	Fail
7	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 1.0 seconds	Pass
8	Assigned Minimum Green Time: 1.0 seconds Actual Minimum Green Time: 5.0 seconds	Fail

Figure 13: Screenshot of a Report Generated by the Minimum Green Phase Timing Test with Fails

- This is the same process for running all of the "Minimum Phase Timing Tests" and the "Maximum Phase Timing Tests."
- Shown in Figure 11, as an example in running one of the single "Min/Max Phase Timing Tests," is Test #9 for checking the Red Min/Max Time of a single phase.

```

Please Choose 'BY NUMBER' an Automated Test to Run: 9
For the Single Min/Max Red Phase Timing Test, Parameters need to be defined:
Phase to be Tested: 8
This Phase's Min/Max Red Time: 0.1
Choose (0) for 'Minimum' or (1) for 'Maximum' Timing Test: 0
Test is Now Running

Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
    
```

Figure 14: Screenshot of a Single Min/Max Phase Timing Test

- After "9" is typed in, Test #9 would prompt for three parameters. Here, Phase 8 is chosen to be tested, followed by the defined minimum red signal time of 0.1 seconds, and finally a zero is typed in to choose that this will be verifying the "Minimum" red time of the chosen phase.
- After the test is completed, in the same fashion as before and for all tests, the report is generated in the same folder to appear as shown in Figure 15.

Single Minimum Red Phase Timing Test

Phases	Resulting Test Information	Pass / Fail
8	Assigned Minimum Red Time: 0.1 seconds Actual Minimum Red Time: 0.1 seconds	Pass

Figure 15: Screenshot of a Report Generated by the Minimum Red Phase Timing Test

Step 4: Configure the ASC into a "Two-Ring Setup" with appropriate phase compatibilities.

- Remove the "Internal ASC Phase Recalls" active on all eight phases.
- This will be the "Controller Sequence Configuration" for the remainder of these tests.
- Note: "Barrier Mode" can simplify this process.
- Cycle power for this change to take effect.
- Run the "Virtual Cabinet" program to clear all internal calls that the ASC applies at startup.

Step 5: Vehicle Call Coordination Test.

- Make sure that the "Detector BIU #1" is enabled under "SDLC Options."
- Choose Test #10 along with desired test parameters.
- Shown in Figure 16, as an example, is the test in which the even phases and then the odd phases are being tested in sequence.

```

Please Choose 'BY NUMBER' an Automated Test to Run: 10
For the Vehicle Call Coordination Test, Parameters need to be defined:
First of Eight Phases to Call: 2
Second Phase to Call: 4
Third Phase to Call: 6
Fourth Phase to Call: 8
Fifth Phase to Call: 1
Sixth Phase to Call: 3
Seventh Phase to Call: 5
Eighth and Final Phase to Call: 7
Test is Now Running

Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
    
```

Figure 16: Screenshot of the Vehicle Call Coordination Test

- The report generated should appear as shown in Figure 17.

Vehicle Call Coordination Test

Phases	Resulting Test Information	Pass / Fail
2	Actual Phase Order: 2	Pass
4	Actual Phase Order: 4	Pass
6	Actual Phase Order: 6	Pass
8	Actual Phase Order: 8	Pass
1	Actual Phase Order: 1	Pass
3	Actual Phase Order: 3	Pass
5	Actual Phase Order: 5	Pass
7	Actual Phase Order: 7	Pass

Figure 17: Screenshot of a Report Generated by the Vehicle Call Coordination Test

Step 6: Pedestrian (PED) Call Coordination Test.

- Assign the four pedestrian signals to Load Switches 13-16 or 9-12.
- Make sure Phases 2, 4, 6, & 8 have a "WALK" and "PED CLR" time of 1 second.
- Choose Test #11 along with desired test parameters.
- Shown in Figure 18, as an example, is the test running the four PED calls in sequential order.

```

Please Choose 'BY NUMBER' an Automated Test to Run: 11
For the Pedestrian Call Coordination Test, Parameters need to be defined:
First of Four PED Signals to Call: 2
Second PED Signal to Call: 4
Third PED Signal to Call: 6
Fourth and Final PED Signal to Call: 8
Test is Now Running
Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
    
```

Figure 18: Screenshot of the Pedestrian Call Coordination Test

- The report generated should appear as shown in Figure 19.

Pedestrian Call Coordination Test

Phases	Resulting Test Information	Pass / Fail
2	Actual Phase Order: 2	Pass
4	Actual Phase Order: 4	Pass
6	Actual Phase Order: 6	Pass
8	Actual Phase Order: 8	Pass

Figure 19: Screenshot of a Report Generated by the Pedestrian Call Coordination Test

Step 7: Preemption Tests.

- Enable the first 6 preemption plans to have a 10 second duration time with an “X” marked in the corresponding phase vehicle call. ("DWEL VEH" field for example).
- This means that for
 - Plan 1: The “X” would be on Phase 1,
 - Plan 2: The “X” would be on Phase 2, ..., and
 - Plan 6: The “X” would be on Phase 6.
- Choose Test #12 along with desired test parameters.
- Shown in Figure 20, as an example, is the test running all six preemption plans in sequential order.

```
Please Choose 'BY NUMBER' an Automated Test to Run: 12
For the Preemption Call Coordination Test, Parameters need to be defined:
First of Six Preemption Types to Call: 1
Second Preemption Type to Call: 2
Third Preemption Type to Call: 3
Fourth Preemption Type to Call: 4
Fifth Preemption Type to Call: 5
Sixth and Final Preemption Type to Call: 6
Test is Now Running
Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
```

Figure 20: Screenshot of the Preemption Call Coordination Test

- The report generated should appear as shown in Figure 21.

Preemption Call Coordination Test

Phases	Resulting Test Information	Pass / Fail
1	Actual Phase Order: 1 Actual Preemption Plan Duration: 10.1	Pass
2	Actual Phase Order: 2 Actual Preemption Plan Duration: 10.1	Pass
3	Actual Phase Order: 3 Actual Preemption Plan Duration: 10.1	Pass
4	Actual Phase Order: 4 Actual Preemption Plan Duration: 10.1	Pass
5	Actual Phase Order: 5 Actual Preemption Plan Duration: 10.1	Pass
6	Actual Phase Order: 6 Actual Preemption Plan Duration: 10.1	Pass

Figure 21: Screenshot of a Report Generated by the Preemption Call Coordination Test

- Notice there is always one extra message frame sent, via SDLC when it comes to the execution of a preemption plan, allotting for the extra tenth of a second.
- Preemption Priority Test:
 - Choose Test #13. (No Test Parameters Required)
 - Shown in Figure 22, as an example, is what the console displays when running this test.

```
Please Choose 'BY NUMBER' an Automated Test to Run: 13
For the Preemption Priority Test, No Parameters need to be defined.
Test is Now Running
Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
```

Figure 22: Screenshot of the Preemption Priority Test

- The report generated should appear as shown in Figure 23.

Preemption Priority Test

Phases	Resulting Test Information	Pass / Fail
6	Actual Phase Order: 6 Actual Preemption Plan Duration: 1.1	Pass
5	Actual Phase Order: 5 Actual Preemption Plan Duration: 1.1	Pass
4	Actual Phase Order: 4 Actual Preemption Plan Duration: 1.1	Pass
3	Actual Phase Order: 3 Actual Preemption Plan Duration: 1.1	Pass
2	Actual Phase Order: 2 Actual Preemption Plan Duration: 1.1	Pass
1	Actual Phase Order: 1 Actual Preemption Plan Duration: 7.1	Fail

Figure 23: Screenshot of a Report Generated by the Preemption Call Coordination Test

- This test was run on a controller that truncated the final highest priority plan from running its full 10 seconds after trumping Priority Plan 2. This is in BOLD showing the noted error and duration of seven seconds (plus the same extra tenth a second as before.)

Step 8: FLASH Test.

- Make sure your load switch assignments are made with the first eight being vehicle phases and Load Switches 9 through 12 being the "Overlaps."
- Now, in the FLASH section make all the signals red except make the straight approaches on the major street yellow (i.e. Phases 2 & 6.) This is often under the Automatic Flash section sometimes abbreviated as AUT.
- Finally, mark the "Trigger" in the "Load Switch FLASH" section (ex. abbr. TGR) for Switches 1, 2, 5, 6, 9, & 11 with an enabling "X".
- Choose Test #14. (No Test Parameters Required)
- Shown in Figure 24, as an example, is the test being run.

```
Please Choose 'BY NUMBER' an Automated Test to Run: 14
For the ASC FLASH Test, No Parameters need to be defined.
Test is Now Running
Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
```

Figure 24: Screenshot of the ASC Flash Test

- The report generated should appear as shown in Figure 25.

ASC FLASH Test

Phases	Resulting Test Information	Pass / Fail
All	ASC FLASH Sequence Successfully Executed	Pass

Figure 25: Screenshot of a Report Generated by the ASC Flash Test

Step 9: Overlap Test.

- Set the load switch assignment for Load Switch 9 through 12 to be the "Overlaps."
- Now, set first vehicle overlap (Load Switch 9 or A) to have Phases 2 & 5 included, the second (Load Switch 10 or B) to have Phases 4 & 7 included, the third (Load Switch 11 or C) to have Phases 1 & 6 included, and the fourth (Load Switch 12 or D) to have Phases 3 & 8 included.
- Choose Test #15. (No Test Parameters Required)
- Shown in Figure 26, as an example, is the test being run.

```
Please Choose 'BY NUMBER' an Automated Test to Run: 15
For the Overlap Configuration Test, No Parameters need to be defined.
Test is Now Running
Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
```

Figure 26: Screenshot of the Overlap Configuration Test

- The report generated should appear as shown in Figure 27.

Overlap Configuration Test

Phases	Resulting Test Information	Pass / Fail
9	Green Time: 50 SDLC messages Yellow Time: 30 SDLC messages	Pass
10	Green Time: 50 SDLC messages Yellow Time: 30 SDLC messages	Pass
11	Green Time: 50 SDLC messages Yellow Time: 30 SDLC messages	Pass
12	Green Time: 50 SDLC messages Yellow Time: 30 SDLC messages	Pass

Figure 27: Screenshot of a Report Generated by the Overlap Configuration Test

Step 10: Channel Mapping Test.

- This test allows for the PED load switch assignments to be before or after the Overlaps.
- Choose Test #16. (No Test Parameters Required)
- Shown in Figure 28, as an example, is the test being run.

```
Please Choose 'BY NUMBER' an Automated Test to Run: 16
For the Channel Mapping Test, No Parameters need to be defined.
Test is Now Running
Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
```

Figure 28: Screenshot of the Channel Mapping Test

- The report generated should appear as shown in Figure 29.

Channel Mapping Test

Phases	Resulting Test Information	Pass / Fail
1	Channel: 1 'Vehicle'	Pass
2	Channel: 2 'Vehicle'	Pass
3	Channel: 3 'Vehicle'	Pass
4	Channel: 4 'Vehicle'	Pass
5	Channel: 5 'Vehicle'	Pass
6	Channel: 6 'Vehicle'	Pass
7	Channel: 7 'Vehicle'	Pass
8	Channel: 8 'Vehicle'	Pass
9	Channel: 9 'Overlap'	Pass
10	Channel: 10 'Overlap'	Pass
11	Channel: 11 'Overlap'	Pass
12	Channel: 12 'Overlap'	Pass
13	Channel: 13 'Pedestrian'	Pass
14	Channel: 14 'Pedestrian'	Pass
15	Channel: 15 'Pedestrian'	Pass
16	Channel: 16 'Pedestrian'	Pass

Figure 29: Screenshot of a Report Generated by the Channel Mapping Test

Step 11: Flashing Yellow Test.

- Set the load switch assignment for Load Switch 9 through 12 to be "Overlaps" and Load Switch 13 through 16 to be "PED."
- Enable the Flashing Yellow Operation to be driven by the overlap or the pedestrian load switches. Make sure the proper protected and permissive phases are set as specified by the NEMA Standard.
- Choose Test #17 along with the desired test parameter.
- Shown in Figure 30, as an example, is the test being run with "PED Driven Flashing Yellow Operation".

```

Please Choose 'BY NUMBER' an Automated Test to Run: 17
For the Flashing Yellow Test, One Parameter needs to be defined:
Type '0' for Overlap-Driven or '1' for PED-Driven Flashing Yellow Operation: 1
Test is Now Running
Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
    
```

Figure 30: Screenshot of the Flashing Yellow Test

- The report generated should appear as shown in Figure 31.

Flashing Yellow Test

Phases	Resulting Test Information	Pass / Fail
1	Flashing Yellow time: 1.0 seconds	Pass
1	Solid Yellow time: 3.0 seconds	Pass
3	Flashing Yellow time: 1.0 seconds	Pass
3	Solid Yellow time: 3.0 seconds	Pass
5	Flashing Yellow time: 1.0 seconds	Pass
5	Solid Yellow time: 3.0 seconds	Pass
7	Flashing Yellow time: 1.0 seconds	Pass
7	Solid Yellow time: 3.0 seconds	Pass

Figure 31: Screenshot of a Report Generated by the Flashing Yellow Test

Step 12: Stop Time Test.

- Make sure "SDLC STOP TIME" is enabled under the settings in the "SDLC PORT 1 CONFIGURATION" menu of the ASC.
- Choose Test #18. (No Test Parameters Required)
- Shown in Figure 32, as an example, is the test being run.

```
Please Choose 'BY NUMBER' an Automated Test to Run: 18
For the Stop Time Test, No Parameters need to be defined.
Test is Now Running
Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
```

Figure 32: Screenshot of the Stop Time Test

- The report generated should appear as shown in Figure 33.

Stop Time Test

Phases	Resulting Test Information	Pass / Fail
All	ASC Stop Time Successfully Executed	Pass

Figure 33: Screenshot of a Report Generated by the Stop Time Test

Step 13: ASC Phase Startup Test.

- Make sure "MMU TO CU SDLC EXTERNAL START" is enabled under the settings in the "SDLC PORT 1 CONFIGURATION" menu of the ASC.
 - Note: Be sure to disable this after running the test.
- Choose Test #19. (No Test Parameters Required)
- Shown in Figure 34, as an example, is the test being run.

```

Please Choose 'BY NUMBER' an Automated Test to Run: 19
For the ASC Phase Startup Test, No Parameters need to be defined.
Test is Now Running
Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
    
```

Figure 34: Screenshot of the ASC Phase Startup Test

- The report generated should appear as shown in Figure 35.

Phases	Resulting Test Information	Pass / Fail
All	ASC Phase Startup Sequence Successfully Executed.	Pass

Figure 35: Screenshot of a Report Generated by the ASC Phase Startup Test

- Remember to disable “MMU TO CU SDLC EXTERNAL START”!

Step 14: Coordination Test.

- Make sure the ASC is put into an action plan running the following coordination pattern:
 - Use Split Pattern: 1
 - Split Sums and the Cycle Time: 40 seconds.

- Offset Value: 0 seconds.
- Actuated Coordination: YES
- Actuated Walk Rest & Phase Re-service: NO
- Split Preference Phases: (Values for Split Fields per Phase)
 - Phase 1 = 5,
 - Phase 2 = 20,
 - Phase 3 = 5,
 - Phase 4 = 10,
 - Phase 5 = 5,
 - Phase 6 = 20,
 - Phase 7 = 5,
 - Phase 8 = 10
- Choose Test #20. (No Test Parameters Required)
- Shown in Figure 36, as an example, is the test being run.

```

Please Choose 'BY NUMBER' an Automated Test to Run: 20
For the Coordination Test, No Parameters need to be defined.
Test is Now Running
Test Complete, Reference Report in Folder: 'ASC_Automated_Tests' for Results.
  
```

Figure 36: Screenshot of the Coordination Test

- The report generated should appear as shown in Figure 37.

FDOT-Traffic Engineering Research Lab	Automated ASC Testing Reports	
<h2>Coordination Test</h2>		
Phases	Resulting Test Information	Pass / Fail
All	ASC Coordination Successfully Implemented	Pass

Figure 37: Screenshot of a Report Generated by the Coordination Test

Step 15: Reflection

- Take a moment to appreciate the old ways of doing things and jump for joy that you didn't have to use a Suitcase Tester!
- You're welcome.

Addendum

The description of what is going on for each of the automated tests is given in the following.

- Test #1 - Test #9 “Phase Timing”:
 - These programs all validate the minimum and maximum phase times. The ASC is to be in a "Single Ring Configuration" with internal "Vehicle Recall" on all phases for each of these tests. The Green, Yellow, and Red Maximum/Minimum Phase Timing Tests allow the user to specify signal times per phase and to monitor the actual illumination time of each for a comparison. The Single Min (or Max) Phase Timing Tests function in the same way, but enable the user to specify a specific phase to test rather than waiting for all eight to run. This is ideal for testing phases of interest when running high phase time counts; ex. 255 seconds (4.25 minutes). Upon completion, the Rich Text Format (RTF) file is created showing the results from the test.
- Test #10 “Vehicle Call Coordination”:
 - This program verifies vehicle calls are serviced properly and in the order they are placed. Execution runs for the eight user-specified phases (1 through 8) verifying vehicle calls are made and serviced in the order they are placed. Upon completion, the RTF file is created showing the results from the test.
- Test #11 “Pedestrian Call Coordination”:
 - This program verifies pedestrian calls are serviced properly and in the order they are placed. The user passes in four parameters specifying four desired pedestrian phases (2, 4, 6, or 8) to service these pedestrian calls in the order passed. Upon completion, the RTF file is created showing the results from the test.
- Test #12 “Preemption Call Coordination”:
 - This program allows the user to pass in any order of the first six different types of preemption calls and verifies they are serviced in that order. Simplified preemption plans of having each priority run its corresponding phase number for 10 seconds are configured in the controller first for validation of their execution. For example, priority one runs only Phase One for ten seconds. An extra message was found each time to get the plans started; hence the 10.1. Upon completion, the RTF file is created showing the results from the test.
- Test #13 “Preemption Priority”:
 - This program essentially runs Preemption calls one after another in descending order beginning with priority 6. In doing this, the program monitors and verifies each higher priority trumps the lower after a second of running; plus the message for starting the plan. The Highest Priority is then allowed to run until completed. The currently tested controller failed the test due to running this plan 3 seconds less than programmed after trumping the preemption plan with the lower priority 2. Upon completion, the RTF file is created showing the results from the test.
- Test #14 “FLASH”:

- With the appropriately defined ASC FLASH sequence programmed into the controller, this program executes the sequence by ceasing replies from one of the required-reply messages of the MMU (Malfunction Management Unit.) This directly causes the ASC to throw up its SDLC fault flag and drive this FLASH sequence through the load switches. This sequence is then monitored for proper functionality and correctness. Upon completion, the RTF file is created showing the results from the test.
- Test #15 “Overlap”:
 - The user needs only to configure the ASC to run minimum green and yellow timing plans for the main eight phases. This program then has Overlaps nine through twelve tested to run in a constant green state across its two enabling phases (Even though one will time out before the other) and then return to its red state accordingly. The final report created shows that exactly 50 SDLC messages (Sent every 1/10 a second) were sent for the appropriate overlap phase to run green for 5 seconds and then 30 SDLC messages for it to run yellow for the defined minimum yellow time of 3 seconds. Upon completion, the RTF file is created showing the results from the test.
- Test #16 “Channel Mapping”:
 - This program verifies the Channel Mapping Configurations seen on the SDLC Bus match those of the user sets via the input panel of the ASC. This program is executed using the vehicle and pedestrian call functionalities. Upon completion, the RTF file is created showing the results from the test.
- Test #17 “Flashing Yellow”:
 - This program monitors the Flashing and Solid Yellow times for phases of desired functionality with a focus on their timing compared to the compatible Green and Yellow phases. The test is designed for a Two Ring implementation. This test will require one input from the user denoting whether the flashing yellows are to be driven from channels 9-12 (input ‘0’) or channels 13-16 (input ‘1’). This will not only account for overlap or pedestrian driven flashing yellows, but also either allowable channel configuration options for both. Upon completion, the RTF file is created showing the results from the test.
- Test #18 “Stop Time Test”:
 - This program places calls to the ASC and then implements an external “Stoptime” signal to freeze the controller’s current state. It is then verified all activity and timing is properly resumed once released. An additional “Stoptime” signal is sent specific to Ring 1 to verify it holds phase timing while Ring 2 advances one phase and proceeds to wait on it. Upon completion, the RTF file is created showing the results from the test.
- Test #19 “ASC Phase Startup”:
 - This program sends the SDLC driven “Ext. Start” signal to reboot the phase operations of the ASC. Upon the reception of this signal, it is then verified that all phases are cycled through. Upon completion, the RTF file is created showing the results from the test. Specific phases will be shown in bold if they are not executed.

- Test #20 “Coordination”:
 - This program brings the controller out of a faulted state and allows for the Local and System cycle times to synchronize with one another. Then, it is verified that a pair of calls are handled on a following cycle, while another set of calls made during the handling of these previous calls are handled immediately after them. Finally, another set of vehicle calls are placed as soon as the controller returns to the major streets to show how they have to wait the longest period of time to be serviced on the next cycle. Upon completion, the RTF file is created showing the results from the test.