	Technical	Report	Documentation	Page
--	-----------	--------	----------------------	------

			i cumuai K	cport Documentation r age
1. Report No. FHWA/TX-07/0-5003-2	2. Government Accession	No.	3. Recipient's Catalog No).
4. Title and Subtitle TESTING FOR COMPLIANCE TO NTCIP STANDA		RDS	5. Report Date September 2006 Resubmitted: Fet Published: April 6. Performing Organizati	oruary 2007 2007 on Code
7. Author(s) Pohort Do Pocho			8. Performing Organizati	on Report No.
9. Performing Organization Name and Address Texas Transportation Institute			10. Work Unit No. (TRA)	[S)
The Texas A&M University System College Station, Texas 77843-3135	n		11. Contract or Grant No. Project 0-5003	
12. Sponsoring Agency Name and Address Texas Department of Transportation Research and Technology Implementation Office			13. Type of Report and Po Technical Report September 2005- 14. Sponsoring Agency C	eriod Covered : August 2006
Austin, Texas 78763-5080				
 Project performed in cooperation with the Texas Department of Transportation and the Federal Highway Administration. Project Title: Development of TxDOT Procedures and Specifications for Testing Device Compliance to NTCIP Standards URL: http://tti.tamu.edu/document/0-5003-2.pdf 			ral Highway mpliance to	
 ^{16. Abstract} The objectives of this two-year project are to define a framework for testing conformance to Nati Transportation Communications for ITS Protocol (NTCIP) standards, identify an approach to des extent to which testing is needed, and recommend the appropriate documentation for such testing To meet the objectives, the first year's report included a summary of past and current efforts by v groups and organizations, a description of available testing tools, and the results of a survey unde understand Texas Department of Transportation's (TxDOT's) testing process and needs. These to followed by discussions of the steps involved in conformance testing, how NTCIP requirements a specified, current TxDOT testing processes, reporting results, and the mapping of requirements to first year's report concluded with an enumerated list of recommendations to establish a testing fra This second year report looks at the details of testing documentation, provides estimates for devel procedures for the various NTCIP-conformant field devices, discusses how to apply the procedure TxDOT testing processes, and presents an outline for training classes. The main portion of the report. 		National o describe the sting activities. by various undertaken to ese topics were ents are ents to tests. The ng framework. developing test cedures to the he report		
concludes with some additional recommendations to establish a testing framework. Appendices addr modifications to TxDOT Closed Circuit Television (CCTV) specifications, a template for a TxDOT specification listing CCTV NTCIP requirements, a set of CCTV test procedures, test results reportin miscellaneous communications test procedures, and a preliminary set of traffic signal controller proc		ces address TxDOT reporting, iller procedures.		
17. Key Words 18. Distribution Statement NTCIP, ITS, Testing No restrictions. This document is available public through NTIS: National Technical Information Service Springfield, Virginia 22161 http://www.ntis.gov		vailable to the		
19. Security Classif.(of this report) Unclassified	20. Security Classif.(of the Unclassified	s page)	21. No. of Pages 494	22. Price

Unclassified	Unclassified
Form DOT F 1700.7 (8-72) Reproduction of complet	ed page authorized

TESTING FOR COMPLIANCE TO NTCIP STANDARDS

by

Robert De Roche Senior Research Specialist Texas Transportation Institute

Report 0-5003-2 Project 0-5003 Project Title: Development of TxDOT Procedures and Specifications for Testing Device Compliance to NTCIP Standards

> Performed in cooperation with the Texas Department of Transportation and the Federal Highway Administration

> > September 2006 Resubmitted: February 2007 Published: April 2007

TEXAS TRANSPORTATION INSTITUTE The Texas A&M University System College Station, Texas 77843-3135

DISCLAIMER

This research was performed in cooperation with the Texas Department of Transportation (TxDOT) and the Federal Highway Administration (FHWA). The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official view or policies of the FHWA or TxDOT. This report does not constitute a standard, specification, or regulation. This report is not intended for construction, bidding, or permit purposes. The research supervisor in charge of the project was Robert De Roche.

The United States Government and the State of Texas do not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the subject of the report.

ACKNOWLEDGMENTS

The research of this project was performed under a cooperative program between the Texas Transportation Institute (TTI), the Texas Department of Transportation, and the Federal Highway Administration. The author wishes to thank the Project Monitoring Committee. Ms. Carol Rawson served as the program coordinator, Mr. Fabian Kalapach served as the project director, and Messrs. Charlie Farnham, David Danz, and Steve Barnettt served as the project advisors. Mr. Wade Odell also served TxDOT research engineer.

Within TTI, the author wishes to thank Dr. Kevin Balke for reviewing and commenting on the report and Mr. Jeremy Johnson for his invaluable help in developing scripts.

TABLE OF CONTENTS

List of Tables xiii List of Abbreviations and Symbols xiv Glossary xviii Chapter 1: Introduction 1 Project Objectives 1 Scope of Project 2 Organization of This Report 2 Chapter 2: NTCIP Testing Documentation 5 Introduction 5 CCTV Documentation 6 Special Specification CCTV Field Equipment 7 Initial Requirements Traceability Matrix 8 Special Specification NTCIP for CCTV Equipment 14 Expanded Requirements Traceability Matrix 20 CCTV Test Procedures 31 CCTV Test Procedures 31 CCTV Test Results 32 Communications Level Test Procedures 35 Application Level. 36 Subnetwork Level. 36 Subnetwork Level. 36 Traffic Signal Controller Documentation. 37 Test Plan and Documentation. 37 Test Plan and Documentation. 37 Test Plan and Documentation. 36 Introduction 36	List of Figures	xii
List of Abbreviations and Symbols xiv Glossary xviii Chapter 1: Introduction 1 Project Objectives 1 Scope of Project 2 Organization of This Report 2 Chapter 2: NTCIP Testing Documentation 5 Introduction 5 CCTV Documentation 6 Special Specification CCTV Field Equipment 7 Initial Requirements Traceability Matrix 8 Special Specification NTCIP for CCTV Equipment 14 Expanded Requirements Traceability Matrix 20 CCTV Test Procedures 31 CCTV Test Results 32 Communications Level Test Procedures 35 Application Level 36 Subnetwork Level 36 Traffic Signal Controller Documentation 36 Traffic Signal Controller Test Procedures 36 Initial Requirements fraceability Matrix 37 Test Plan and Documentation 36 Traffic Signal Controller Test Procedures 60 Traffic Signal Controller Test Results 60 Detector Requirements 61	List of Tables	xiii
Glossary xviii Chapter 1: Introduction 1 Project Objectives 1 Scope of Project. 2 Organization of This Report. 2 Chapter 2: NTCIP Testing Documentation. 5 Introduction 5 CCTV Documentation. 6 Special Specification CCTV Field Equipment 7 Initial Requirements Traceability Matrix 8 Special Specification NTCIP for CCTV Equipment 14 Expanded Requirements Traceability Matrix 20 CCTV Test Procedures 31 CCTV Test Procedures 31 CCTV Test Results 32 Communications Level Test Procedures 35 Transport Level 36 Subnetwork Level 36 Subnetwork Level 36 Traffic Signal Controller Documentation 36 Traffic Signal Controller Test Procedures 60 Traffic Signal Controller Test Procedures 60 Traffic Signal Controller Test Results 60 Detector Requirements 61 Chapter 3: Developing Additional Test Procedures 63	List of Abbreviations and Symbols	xiv
Chapter 1: Introduction 1 Project Objectives 1 Scope of Project 2 Organization of This Report 2 Chapter 2: NTCIP Testing Documentation 5 Introduction 5 CCTV Documentation 6 Special Specification CCTV Field Equipment 7 Initial Requirements Traceability Matrix 8 Special Specification NTCIP for CCTV Equipment 14 Expanded Requirements Traceability Matrix 20 CCTV Test Procedures 31 CCTV Test Procedures 32 Communications Level Test Procedures 35 Application Level 36 Subnetwork Level 36 Traffic Signal Controller Documentation 36 Traffic Signal Controller Documentation 36 Traffic Signal Controller Test Procedures 36 Initial Requirements Traceability Matrix 37 Test Plan and Documentation 36 Traffic Signal Controller Test Results 60 Detector Requirements 61 Chapter 3: Developing Additional Test Procedures 63 Introduction 6	Glossary	xviii
Chapter 1: Introduction 1 Project Objectives 1 Scope of Project. 2 Organization of This Report. 2 Chapter 2: NTCIP Testing Documentation. 5 Introduction 5 Introduction 6 Special Specification CCTV Field Equipment 7 Initial Requirements Traceability Matrix 8 Special Specification NTCIP for CCTV Equipment 14 Expanded Requirements Traceability Matrix 20 CCTV Test Procedures 31 CCTV Test Procedures 31 CCTV Test Results 32 Communications Level Test Procedures 35 Application Level. 36 Subnetwork Level. 36 Subnetwork Level. 36 Traffic Signal Controller Documentation. 36 Traffic Signal Controller Test Procedures 60 Traffic Signal Controller Test Procedures 60 Traffic Signal Controller Test Procedures 61 Chapter 3: Developing Additional Test Procedures 61 Chapter 3: Developing Additional Test Procedures 63 Introduction 63		
Project Objectives 1 Scope of Project. 2 Organization of This Report. 2 Chapter 2: NTCIP Testing Documentation. 5 Introduction 5 CCTV Documentation. 6 Special Specification CCTV Field Equipment 7 Initial Requirements Traceability Matrix 8 Special Specification NTCIP for CCTV Equipment 14 Expanded Requirements Traceability Matrix 20 CCTV Test Procedures 31 CCTV Test Procedures 32 Communications Level Test Procedures 35 Application Level. 36 Subnetwork Level 36 Subnetwork Level 36 Traffic Signal Controller Documentation. 36 Traffic Signal Controller Documentation. 36 Traffic Signal Controller Test Procedures 36 Initial Requirements Traceability Matrix 37 Test Plan and Documentation 59 Traffic Signal Controller Test Results 60 Detector Requirements 61 Chapter 3: Developing Additional Test Procedures 63 Introduction	Chapter 1: Introduction	1
Scope of Project. 2 Organization of This Report. 2 Organization of This Report. 2 Chapter 2: NTCIP Testing Documentation. 5 Introduction 5 CCTV Documentation. 6 Special Specification CCTV Field Equipment 7 Initial Requirements Traceability Matrix. 8 Special Specification NTCIP for CCTV Equipment 14 Expanded Requirements Traceability Matrix 20 CCTV Test Procedures 31 CCTV Test Procedures 32 Communications Level Test Procedures 35 Application Level 36 Subnetwork Level 36 Traffic Signal Controller Documentation 36 TxDOT Specifications for Traffic Signal Controllers 36 Initial Requirements Traceability Matrix 37 Test Plan and Documentation 59 Traffic Signal Controller Test Procedures 60 Detector Requirements 61 Chapter 3: Developing Additional Test Procedures 63 Introduction 63 Introduction 63 Introduction 64	Project Objectives	1
Organization of This Report. 22 Chapter 2: NTCIP Testing Documentation. 5 Introduction 5 CCTV Documentation 6 Special Specification CCTV Field Equipment 7 Initial Requirements Traceability Matrix 8 Special Specification NTCIP for CCTV Equipment 14 Expanded Requirements Traceability Matrix 20 CCTV Test Procedures 31 CCTV Test Procedures 32 Communications Level Test Procedures 35 Application Level 36 Subnetwork Level 36 Traffic Signal Controller Documentation 36 Traffic Signal Controller Documentation 36 Traffic Signal Controller Test Procedures 36 Initial Requirements Traceability Matrix 37 Test Plan and Documentation 59 Traffic Signal Controller Test Procedures 60 Traffic Signal Controller Test Procedures 60 Detector Requirements 61 Chapter 3: Developing Additional Test Procedures 63 Introduction 63 Introduction 63 Introduction	Scope of Project	2
Chapter 2: NTCIP Testing Documentation. 5 Introduction 5 CCTV Documentation 6 Special Specification CCTV Field Equipment 7 Initial Requirements Traceability Matrix. 8 Special Specification NTCIP for CCTV Equipment 14 Expanded Requirements Traceability Matrix 20 CCTV Test Procedures 31 CCTV Test Procedures 32 Communications Level Test Procedures 35 Application Level. 35 Transport Level. 36 Subnetwork Level. 36 Subnetwork Level. 36 Taffic Signal Controller Documentation 36 Traffic Signal Controller Documentation 36 Initial Requirements Traceability Matrix. 37 Test Plan and Documentation 59 Traffic Signal Controller Test Procedures 60 Traffic Signal Controller Test Results 60 Detector Requirements 61 Chapter 3: Developing Additional Test Procedures 63 Introduction 63 ITS Field Device Estimates 64 NTCIP 103-TMP 64 <td>Organization of This Report</td> <td>2</td>	Organization of This Report	2
Introduction5CCTV Documentation.6Special Specification CCTV Field Equipment.7Initial Requirements Traceability Matrix.8Special Specification NTCIP for CCTV Equipment.14Expanded Requirements Traceability Matrix.20CCTV Test Procedures.31CCTV Test Procedures.32Communications Level Test Procedures.35Application Level.35Transport Level.36Subnetwork Level.36Traffic Signal Controller Documentation.36TxADOT Specifications for Traffic Signal Controllers.36Initial Requirements Traceability Matrix.37Test Plan and Documentation.59Traffic Signal Controller Test Procedures.60Traffic Signal Controller Test Procedures.60Traffic Signal Controller Test Procedures.61Chapter 3: Developing Additional Test Procedures.63Introduction.63ITS Field Device Estimates.64NTCIP 103-TMP.64NTCIP 1201-GLO.65	Chapter 2: NTCIP Testing Documentation	
CCTV Documentation	Introduction	5
Special Specification CCTV Field Equipment	CCTV Documentation	6
Initial Requirements Traceability Matrix 8 Special Specification NTCIP for CCTV Equipment 14 Expanded Requirements Traceability Matrix 20 CCTV Test Procedures 31 CCTV Test Results 32 Communications Level Test Procedures 35 Application Level 35 Transport Level 36 Subnetwork Level 36 Traffic Signal Controller Documentation 36 TxDOT Specifications for Traffic Signal Controllers 36 Initial Requirements Traceability Matrix 37 Test Plan and Documentation 59 Traffic Signal Controller Test Procedures 60 Traffic Signal Controller Test Procedures 60 Detector Requirements 61 Chapter 3: Developing Additional Test Procedures 63 Introduction 63 ITS Field Device Estimates 64 NTCIP 103-TMP 64 NTCIP 1201-GLO 65	Special Specification CCTV Field Equipment	7
Special Specification NTCIP for CCTV Equipment 14 Expanded Requirements Traceability Matrix 20 CCTV Test Procedures 31 CCTV Test Results 32 Communications Level Test Procedures 35 Application Level 35 Transport Level 36 Subnetwork Level 36 Traffic Signal Controller Documentation 36 TxDOT Specifications for Traffic Signal Controllers 36 Initial Requirements Traceability Matrix 37 Test Plan and Documentation 59 Traffic Signal Controller Test Procedures 60 Traffic Signal Controller Test Procedures 61 Chapter 3: Developing Additional Test Procedures 63 ITS Field Device Estimates 64 NTCIP 1103-TMP 64 NTCIP 1201-GLO 65	Initial Requirements Traceability Matrix	8
Expanded Requirements Traceability Matrix20CCTV Test Procedures31CCTV Test Results32Communications Level Test Procedures35Application Level.35Transport Level36Subnetwork Level36Traffic Signal Controller Documentation.36TxDOT Specifications for Traffic Signal Controllers36Initial Requirements Traceability Matrix37Test Plan and Documentation59Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Results60Detector Requirements61Chapter 3: Developing Additional Test Procedures63Introduction63ITS Field Device Estimates64NTCIP 1103-TMP64NTCIP 1201-GLO65	Special Specification NTCIP for CCTV Equipment	14
CCTV Test Procedures31CCTV Test Results32Communications Level Test Procedures35Application Level35Transport Level36Subnetwork Level36Traffic Signal Controller Documentation36TxDOT Specifications for Traffic Signal Controllers36Initial Requirements Traceability Matrix37Test Plan and Documentation59Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Procedures61Chapter 3: Developing Additional Test Procedures63Introduction63ITS Field Device Estimates64NTCIP 1103-TMP64NTCIP 1201-GLO65	Expanded Requirements Traceability Matrix	20
CCTV Test Results32Communications Level Test Procedures35Application Level.35Transport Level.36Subnetwork Level.36Traffic Signal Controller Documentation.36TxDOT Specifications for Traffic Signal Controllers36Initial Requirements Traceability Matrix.37Test Plan and Documentation59Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Results60Detector Requirements61Chapter 3: Developing Additional Test Procedures63Introduction63ITS Field Device Estimates64NTCIP 1103-TMP64NTCIP 1201-GLO65	CCTV Test Procedures	31
Communications Level Test Procedures35Application Level35Transport Level36Subnetwork Level36Traffic Signal Controller Documentation36TxDOT Specifications for Traffic Signal Controllers36Initial Requirements Traceability Matrix37Test Plan and Documentation59Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Results60Detector Requirements61Chapter 3: Developing Additional Test Procedures63Introduction63ITS Field Device Estimates64NTCIP 1103-TMP64NTCIP 1201-GLO65	CCTV Test Results	32
Application Level.35Transport Level.36Subnetwork Level.36Traffic Signal Controller Documentation.36TxDOT Specifications for Traffic Signal Controllers36Initial Requirements Traceability Matrix.37Test Plan and Documentation59Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Results60Detector Requirements61Chapter 3: Developing Additional Test Procedures63Introduction63ITS Field Device Estimates64NTCIP 1103-TMP64NTCIP 1201-GLO65	Communications Level Test Procedures	35
Transport Level36Subnetwork Level36Traffic Signal Controller Documentation36TxDOT Specifications for Traffic Signal Controllers36Initial Requirements Traceability Matrix37Test Plan and Documentation59Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Results60Detector Requirements61Chapter 3: Developing Additional Test Procedures63Introduction63ITS Field Device Estimates64NTCIP 1103-TMP64NTCIP 1201-GLO65	Application Level	35
Subnetwork Level36Traffic Signal Controller Documentation36TxDOT Specifications for Traffic Signal Controllers36Initial Requirements Traceability Matrix37Test Plan and Documentation59Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Results60Detector Requirements61Chapter 3: Developing Additional Test Procedures63Introduction63ITS Field Device Estimates64NTCIP 1103-TMP64NTCIP 1201-GLO65	Transport Level	36
Traffic Signal Controller Documentation36TxDOT Specifications for Traffic Signal Controllers36Initial Requirements Traceability Matrix37Test Plan and Documentation59Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Results60Detector Requirements61Chapter 3: Developing Additional Test Procedures63Introduction63ITS Field Device Estimates64NTCIP 1103-TMP64NTCIP 1201-GLO65	Subnetwork Level	36
TxDOT Specifications for Traffic Signal Controllers36Initial Requirements Traceability Matrix37Test Plan and Documentation59Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Results60Detector Requirements61Chapter 3: Developing Additional Test Procedures63Introduction1TS Field Device Estimates64NTCIP 1103-TMP64NTCIP 1201-GLO65	Traffic Signal Controller Documentation	36
Initial Requirements Traceability Matrix37Test Plan and Documentation59Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Results60Detector Requirements61Chapter 3: Developing Additional Test Procedures63Introduction1TS Field Device Estimates64NTCIP 1103-TMP64NTCIP 1201-GLO65	TxDOT Specifications for Traffic Signal Controllers	36
Test Plan and Documentation59Traffic Signal Controller Test Procedures60Traffic Signal Controller Test Results60Detector Requirements61Chapter 3: Developing Additional Test Procedures63IntroductionIntroduction63ITS Field Device Estimates64NTCIP 1103-TMP64NTCIP 1201-GLO65	Initial Requirements Traceability Matrix	37
Traffic Signal Controller Test Procedures .60 Traffic Signal Controller Test Results .60 Detector Requirements .61 Chapter 3: Developing Additional Test Procedures .63 Introduction .63 ITS Field Device Estimates .64 NTCIP 1103-TMP .64 NTCIP 1201-GLO .65	Test Plan and Documentation	59
Traffic Signal Controller Test Results .60 Detector Requirements .61 Chapter 3: Developing Additional Test Procedures .63 Introduction .63 ITS Field Device Estimates .64 NTCIP 1103-TMP .64 NTCIP 1201-GLO .65	Traffic Signal Controller Test Procedures	60
Detector Requirements	Traffic Signal Controller Test Results	60
Chapter 3: Developing Additional Test Procedures63Introduction	Detector Requirements	61
Chapter 3: Developing Additional Test Procedures 63 Introduction 63 ITS Field Device Estimates 64 NTCIP 1103-TMP 64 NTCIP 1201-GLO 65		
Introduction	Chapter 3: Developing Additional Test Procedures	63
NTCIP 1201-GLO		
NTCIP 1103-1MP	NTCID 1102 TMD	64
NICIP 1201-GLO	NTCIP 1201 CLO	64
	NTCIP 1201-GLU	
NTCIP 1202-ASC	NTCIP 1202 DMS	03
NTCIP 1203-DWS00		00
NICIT 1204-E55	NICIT 1204-E55 NTCID 1205 CCTV	00
NICIF 1203-001 V	NICH 1203-UCI V	00
NTCID 1200-DUVI00	NICIF 1200-DCWI NTCID 1207 PMC	00
NTCIP 1207-RIVC	NTCIP 1207-KWC	07

NTCIP 1209-TSS	
NTCIP 1210-FMS	
NTCIP 1211-SCP	
NTCIP 1213-ELMS	
Chapter 4: Applying Testing Procedures	
Testing Processes	
Internal TxDOT Testing Process	
Prequalification Testing	
QPL Testing	71
Configuration Testing	72
Sample Environmental System Testing	73
System Testing	73
Contractor Testing Process	
Configuration Management and Version Control	
Chapter 5: Training	
Introduction	77
Audience	77
Training Class Outlines	
Training Class Evaluation Form	
Chapter 6: Recommendations	
Testing Framework	
Future Development	
ELMS Test Procedures	
Generic Database	
References	
Appendix A: Special Specification for CCTV Equipment	
References for Appendix A	
Appendix B: Special Specification - NTCIP for CCTV Equi	nment
	p
References for Appendix B	
Appendix C: CCTV Test Procedures	
Introduction	
Test Case Summary	
Test Cases	
CCTV PRL Information	
Cabinet Alarm	
Enclosure Alarm	
Video Loss Alarm	
Temperature Alarm	
Pressure Alarm	

Local Remote Alarm	126
Washer Fluid Alarm	127
Identify Device	128
Identify Preset Position Range	128
Identify Pan Limits	129
True North Offset	130
Identify Tilt Limits	130
Identify Zoom Limit	131
Identify Focus Limit	131
Identify Iris Limit	132
Identify Pan-Tilt Step Angle Minimum	132
Identify Zone Functions	133
Monitor Discrete Input	133
Monitor Discrete Output	135
Get Availability of Equipment	137
Control Camera Power	137
Control Heater Power	138
Control Wiper	138
Control Washer	139
Control Blower	139
Delta Focus Motion	140
Absolute Focus Motion	141
Continuous Focus Motion with Timeout	
Continuous Focus Motion with Stop	
Retrieve Module Table	.143
Global Set ID.	144
Delta Iris Motion	
Absolute Iris Motion	146
Continuous Iris Motion with Timeout	
Continuous Iris Motion with Stop	148
Get and Set Label	149
Display Camera Location	150
Get Availability of Lens Equipment	151
Control Auto Iris	151
Control Auto Focus	152
Menu	153
Delta Pan Motion	154
Absolute Pan Motion	155
Continuous Pan Motion with Timeout	156
Continuous Pan Motion with Stop	157
Change Administrator Community Name	158
Change User Community Name	159
Delta Tilt Motion	162
Absolute Tilt Motion	163
Continuous Tilt Motion with Timeout	164
Continuous Tilt Motion with Stop	165
▲	

Preset Position	166
Get-Set Zone	
Move In and Out of Zone	
Delta Zoom Motion	
Absolute Zoom Motion	
Continuous Zoom Motion with Timeout	170
Continuous Zoom Motion with Stop	171
References for Appendix C	172
Appendix D: CCTV Protocol Implementation Conformance Specification	173
Introduction	173
Interpreting Results	173
References for Appendix D	190
Appendix E: Communications and Miscellaneous Test Procedures	191
Introduction	191
SNMP Test Cases	191
Transportation Transport Test Cases	195
Point to Multi-Point with RS232 Test Cases	196
STMP Test Cases	
Response Time Test Case	100
Response Time Test Case	
References for Appendix E	201
References for Appendix E	201 203
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction	201 203 203
References for Appendix E	201 203 203 246
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F Appendix G: Traffic Signal Controller Test Procedures	201 203 203 246 247
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F Appendix G: Traffic Signal Controller Test Procedures Introduction	201 203 203 246 247 247
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations	201 203 203 203 246 247 247 247
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations Test Case Summary	201 203 203 246 247 247 247 247 249
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F. Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations. Test Case Summary. Test Cases	201 203 203 246 247 247 247 247 247 249 254
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F. Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations Test Case Summary Test Cases ASC PRL Information	201 203 203 246 247 247 247 247 247 247 247 249 254 254
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F. Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations. Test Case Summary. Test Cases. ASC PRL Information Four-Phase Diamond Sequencing	201 203 203 246 247 247 247 247 247 247 247 247 247
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations Test Case Summary Test Cases ASC PRL Information Four-Phase Diamond Detector Operations	201 203 203 246 247 247 247 247 247 247 247 247 247 247
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations Test Cases ASC PRL Information Four-Phase Diamond Sequencing Four-Phase Diamond Detector Operations Detector 1 Operations	201 203 203 246 247 247 247 247 247 247 249 254 254 255 283 283
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations Test Case Summary Test Cases ASC PRL Information Four-Phase Diamond Detector Operations Detector 1 Operations Detector 2 Operations	201 203 203 246 247 247 247 247 247 249 254 254 254 255 283 283 283
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations Test Case Summary Test Cases ASC PRL Information Four-Phase Diamond Sequencing Four-Phase Diamond Detector Operations Detector 1 Operations Detector 2 Operations Detector 3 Operations	201 203 203 246 247 247 247 247 247 247 247 247 247 247 247 245 254 255 283 283 288 298
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations Test Cases ASC PRL Information Four-Phase Diamond Sequencing Four-Phase Diamond Detector Operations Detector 1 Operations Detector 2 Operations Detector 3 Operations Detector 4 Operations	201 203 203 246 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 248 283 283 288 298 309
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations Test Cases ASC PRL Information Four-Phase Diamond Sequencing Four-Phase Diamond Detector Operations Detector 1 Operations Detector 2 Operations Detector 3 Operations Detector 4 Operations Detector 5 Operations	201 203 203 246 247 247 247 247 247 247 247 247 247 247 247 247 247 247 248 255 283 283 283 288 298 309 320
References for Appendix E Appendix F: Traffic Signal Controller Test Documentation Introduction References for Appendix F Appendix G: Traffic Signal Controller Test Procedures Introduction Detector Operations Test Cases ASC PRL Information Four-Phase Diamond Sequencing Four-Phase Diamond Detector Operations Detector 1 Operations Detector 2 Operations Detector 3 Operations Detector 4 Operations Detector 5 Operations Detector 6 Operations	201 203 203 246 247 247 247 247 247 247 247 247 247 247 247 247 247 247 247 248 255 283 288 288 298 309 320 324

Detector 8 Operations	
Detector 9 Operations	
Detector 10 Operations	
Detector 11 Operations	
Detector 12 Operations	
Detector 13 Operations	
Detector 14 Operations	
Detector 15 Operations	
Detector 16 Operations	
Detector 17 Operations	
Detector 18 Operations	
Detector Operations Setup	
Detector Operations Teardown	
Detector Delay	470
References for Appendix G	475

LIST OF FIGURES

Page

Figure 1. CCTV PRL Information Test Case Results	
Figure 2. CCTV Cabinet Alarm Test Case Results Form.	
Figure 3. Test Results Indicating Critical Results.	61
Figure 4. TxDOT Testing Activities.	70
Figure 5. Contractor Testing Activities.	74
Figure 6. Instrumentation Testing Tool Information Example	75

LIST OF TABLES

Page

Table 1. Requirements Traceability Matrix with NTCIP Objects.	9
Table 2. Requirements Traceability Matrix from NTCIP 1203-DMS	
Table 3. Requirements Traceability Matrix with Test Procedures.	
Table 4. TSC Requirements Traceability Matrix with Test Procedures	
Table 5. NTCIP Standard Statistics and Test Procedure Efforts.	
Table C-1. CCTV Test Case Summary	
Table E-1. SNMP Test Case Summary	
Table E-2. Transportation Transport Test Case Summary	
Table E-3. PMPP with RS-232 Test Case Summary	
Table E-4. STMP Test Case Summary	
Table E-5. Response Time Test Case Summary	
Table G-1. Traffic Signal Controller Test Case Summary	
Table G-2. Additional Traffic Signal Controller Test Case Summary	

LIST OF ABBREVIATIONS AND SYMBOLS

AASHTO	American Association of State Highway and Transportation Officials
ASC	Actuated Signal Controller
ATMS	Advanced Transportation Management System
BER	Basic Encoding Rules
CCTV	Closed Circuit Television
СНАР	Challenge Handshake Authentication Protocol
CRC	Cyclic Redundancy Check
DMS	Dynamic Message Sign
DMS 11170-TSC	DMS-11170, Fully Actuated, Solid-State Traffic Signal Controller
	Assembly
DOT	Department of Transportation
DUT	Device Under Test
EIA	Electronics Industry Alliance
ELMS	Electrical and Lighting Management Systems
ESS	Environmental Sensor Station
FDOT	Florida Department of Transportation
FHWA	Federal Highway Administration
FMS	Field Management Station
FTP	File Transfer Protocol
HAR	Highway Advisory Radio
HDLC	High-Level Data Link Control
HITL	Hardware-in-the-Loop
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPI	Initial Protocol Identifier
ISO	International Organization for Standardization
ITE	Institute of Transportation Engineers
ITL	Interoperability Test Lab
ITS	Intelligent Transportation Systems

MIB	Management Information Base
NEMA	National Electrical Manufacturers Association
NIATT	National Institute for Advanced Transportation Technology
NTCIP	National Transportation Communications for ITS Protocol
NTCIP 1103-TMP	NTCIP 1103 – Transportation Management Protocols
NTCIP 1201-GLO	NTCIP 1201 – Global Object Definitions
NTCIP 1202-ASC	NTCIP 1202 – Object Definitions for Actuated Traffic Signal Controller
	Units
NTCIP 1203-DMS	NTCIP 1203 – Object Definitions for Dynamic Message Signs (DMS)
NTCIP 1204-ESS	NTCIP 1204 – Environmental Sensor Station Interface Standard
NTCIP 1205-CCTV	NTCIP 1205 – Object Definitions for Closed Circuit Television (CCTV)
	Camera Control
NTCIP 1206-DCM	NTCIP 1206 – Object Definitions for Data Collection and Monitoring
	(DCM) Devices
NTCIP 1207-RMC	NTCIP 1207 – Object Definitions for Ramp Meter Control (RMC) Units
NTCIP 1208-SW	NTCIP 1208 – Object Definitions for Closed Circuit Television (CCTV)
	Switching
NTCIP 1209-TSS	NTCIP 1209 – Data Element Definitions for Transportation Sensor
	Systems
NTCIP 1210-FMS	NTCIP 1210 – Field Management Stations - Part 1: Object Definitions
	for Signal System Masters
NTCIP 1211-SCP	NTCIP 1211 – Object Definitions for Signal Control and Prioritization
NTCIP 1213-ELMS	NTCIP 1213 – Objects Definitions for Electrical and Lighting
	Management Systems
NTCIP 2101-PMPP/RS232	NTCIP 2101 – Point to Multi-Point Protocol Using RS-232 Subnetwork
	Profile
NTCIP 2102-PMPP/FSK	NTCIP 2102 – Point to Multi-Point Protocol Using FSK Modem
	Subnetwork Profile
NTCIP 2103-PPP	NTCIP 2103 – Point-to-Point Protocol over RS-232 Subnetwork Profile
NTCIP 2104-Ethernet	NTCIP 2104 – Ethernet Subnetwork Profile
NTCIP 2201-T2	NTCIP 2201 – Transportation Transport Profile

NTCIP 2202-ITP	NTCIP 2202 – Internet (TCP/IP and UDP/IP)Transport Profile			
NTCIP 2301-STMF	NTCIP 2301 – Simple Transportation Management Framework			
	Application Profile			
NTCIP 8007-TEST	NTCIP 8007 – Testing and Conformity Assessment Documentation			
	within NTCIP Standards Publications			
NTCIP 9012-TG	NTCIP 9012 – Testing and Conformity Assessment User Guide for			
	NTCIP Field Devices and Center-to-Field Communications			
NTCIP Guide	NTCIP 9001 – The NTCIP Guide			
OER	Octet Encoding Rules			
OID	Object Identifier			
PDF	Portable Document Format			
PDU	Protocol Data Unit			
PICS	Protocol / Profile Implementation Conformance Specification			
PMPP	Point to Multi-Point Protocol			
PPP	Point-to-Point Protocol			
PRL	Protocol / Profile Requirements List			
PTZ	Pan, Tilt, and Zoom			
QPL	Qualified Products List			
RFC	Request for Comment			
RS	Recommended Standard			
RTM	Requirements Traceability Matrix			
SFMP	Simple Fixed Message Protocol			
SNMP	Simple Network Management Protocol			
SS 6025	TxDOT 2004 Special Specification 6025 CCTV Field Equipment			
SS 6026	TxDOT 2004 Special Specification 6026 National Transportation			
	Communications for ITS Protocol for Dynamic Message Signs			
SS 6504	TxDOT 1993 Special Specifications 6504 – Testing, Training,			
	Documentation and Warranty			
STMP	Simple Transportation Management Protocol			
TCL	Tool Command Language			
ТСР	Transmission Control Protocol			

TERL	Traffic Engineering Research Lab
TFTP	Trivial File Transfer Protocol
TS	Traffic Section
TTI	Texas Transportation Institute
TxDOT	Texas Department of Transportation
UDP	User Datagram Protocol
UP	Unnumbered Poll
V&V	Validation and Verification
VIVDS	Video Imaging Vehicle Detection Systems

GLOSSARY

Compliance	Compliance is a condition that exists when an item meets all of the requirements of a procurement specification.
Conformance	Conformance is a condition that exists when an item meets all of the mandatory requirements as defined by a formal standard.
Management Application	A generic term for any computer-based software used to configure, control, or monitor the operation of a field device or devices.

CHAPTER 1: INTRODUCTION

PROJECT OBJECTIVES

The objectives of this two-year project are to define a framework for testing conformance to National Transportation Communications for Intelligent Transportation Systems Protocol (NTCIP) standards, identify the approaches used to describe the extent to which testing is needed, and recommend appropriate documentation for such testing activities. This research project will accomplish the following for TxDOT:

- Assist TxDOT in developing a comprehensive approach to testing Intelligent Transportation Systems (ITS) -related hardware and software to ensure conformance with national standards and compliance with TxDOT specifications.
- Identify TxDOT testing needs and available resources to meet those needs.
- Develop a framework, along with methodologies and procedures as needed, for conducting both laboratory and field-testing of devices.
- Assist TxDOT in evaluating options for testing of ITS hardware and software as part of procurement and construction projects.
- Assist TxDOT in developing procedures and reports for documenting the results of the testing program.
- Develop outlines for training courses that convey how to use and interpret the results of the testing program.

The goal of testing is to ensure that the design, implementation, and functionality of a product meet user needs and requirements. NTCIP standards define a set of protocols associated with communications technologies used in transportation-related products. These protocols ensure systems that integrate NTCIP-conformant products can communicate using a common language and describe information in a consistent manner. The goal of NTCIP testing is to ensure that a product follows the protocol rules that define the common language and that the information exchanged by the components of a system is meaningful and understood. The intent of this project is to look at tasks involved and methods used to check conformance to NTCIP. The intention is to look also at a means of integrating NTCIP testing into TxDOT's current testing program.

SCOPE OF PROJECT

The scope of this research is testing of conformance to NTCIP standards and compliance to TxDOT specifications that reference NTCIP standards. NTCIP standards define common methods and protocols that enable ITS devices to communicate. The standards also define the language and words used when communicating. In some cases, no other standards exist that define the meaning of the words or how the words relate to functionality; therefore, some NTCIP standards also define the functionality of an ITS device. This research looks at the types of testing involved in showing conformance to NTCIP standards and compliance to TxDOT specifications, the resources available to accomplish the tasks, the specific needs for testing by TxDOT, the current testing process within TxDOT, and the testing tools available to help in the testing process.

ORGANIZATION OF THIS REPORT

In addition to this introduction chapter, this report contains five chapters and seven appendices. Chapter 2 of this report discusses NTCIP testing documentation. Testing documentation not only includes actual test procedures but also documentation that relates to specifying NTCIP requirements and how testing verifies those requirements. Chapter 2 provides a description of steps involved in creating testing documentation. By way of example, discussion focuses on the full set documentation for CCTV camera controllers beginning with a description of the modifications to TxDOT Special Specification 6025 (SS 6025) to reference NTCIP requirements (*1*). Appendix A provides suggested changes to that specification. Since TxDOT uses a separate document to define NTCIP requirements, a discussion of the content for a new document titled NTCIP for CCTV Equipment ensues. This document cites what NTCIP standards, what mandatory and optional conformance groups within the standards, and what optional object definitions within the standards to reference. Appendix B contains a template for the new TxDOT specification.

Chapter 2 also deals with a Requirements Traceability Matrix (RTM) that correlates the requirements in TxDOT Special Specification 6025 to NTCIP Object Definitions and Test Procedure Identifiers and presents an introduction and general description of the CCTV test procedures. Appendix C of this report contains the actual procedures.

What follows is a discussion on reporting results. The researcher found that the clearest means of reporting results is to use the Protocol/Profile Requirements List (PRL) that appears in the NTCIP standards or the test procedure documentation. Appendix D contains a PRL that shows test results. The CCTV Test Results part of Chapter 2 shows a sample of how a test procedure can show test results.

The next section of Chapter 2 deals with traffic signal controller documentation. TxDOT department material specification DMS-11170, Fully Actuated, Solid-State Traffic Signal Controller Assembly (DMS 11170-TSC) contains NTCIP requirements (2). As such, the researcher makes several suggestions for addressing communications. The next part discusses an initial RTM based upon the requirements in DMS 11170-TSC (2). This is an initial RTM because it only deals with the requirements in DMS 11170-TSC, individual objects definitions from the relevant standards, and the limited number of traffic signal controller test procedures that exist (2). Appendix E provides a description and listing of existing communications test procedures that may be applicable to any NTCIP field device. Appendix F contains a Test Design Specification for addressing all the functions of a traffic signal controller. It uses IEEE Std. 829 – IEEE Standard for Software Test Documentation as a guide on the organization and content (*3*).

Chapter 3 provides estimates for the level of effort needed to develop test procedures for all NTCIP field devices. The estimates use statistics about the content of the NTCIP standards, the researcher's experience in developing test procedures, discussions with NTCIP working group chairs and editors, and current NTCIP development plans.

Chapter 4 deals with applying NTCIP testing procedures in TxDOT testing processes. Discussions deal with TxDOT's internal process consisting of Qualified Products List (QPL), sample environmental, configuration, and system testing. It also deals with TxDOT's external process or contractor testing consisting of design approval, demonstration, stand-alone, and system integration test steps. The discussions focus on how NTCIP testing might apply to each of these processes and steps. Chapter 4 concludes with a brief discussion of configuration management and version control.

Chapter 5 presents two training course outlines. The first looks at testing from an NTCIP perspective, and the second outline examines testing from a TxDOT perspective. The first explores the difference between conformance and compliance testing. Conformance testing

applies to the NTCIP standards, and compliance testing applies to TxDOT specifications. The outline suggests that the training first look at background information and the two types of standards: data dictionaries and protocols/profiles. It would then cover how to view the NTCIP framework in order to understand how individual standards make up an actual implementation. Topics on terminology and techniques and on how best to interpret and report the results follow.

The second outline in Chapter 5 looks at testing from the TxDOT perspective. It deals with the how to balance a desire to fully test an implementation with the reality issues of lack of resources and time. A discussion of risk management looks at how to minimize the testing effort and still maintain a high level of confidence. Further topics cover what to test, the techniques to use, and the various testing tools that are available. The chapter concludes with an outline of how to address configuration management and includes a suggested evaluation form.

Chapter 6 of the report adds some additional recommendations on defining a framework for the testing of conformance to NTCIP and integrating it into the current TxDOT testing program. The recommendations are in addition to the 17 recommendations that the first year report included.

CHAPTER 2: NTCIP TESTING DOCUMENTATION

INTRODUCTION

A major component of any testing framework is supporting documentation. There must be specific requirements or specifications, a document that correlates those requirements to a set of test procedures for verifying the requirements, and documented test procedures to ensure that the verification process is consistent. For any organization as large as TxDOT, consistent reporting of results should also be a part of the framework.

By specifying conformance to NTCIP standards, one can assume that TxDOT shares the view the standards promote:

- Compatibility
- Interoperability
- Interchangeability

System components should be compatible so that different components can share a common communications infrastructure. There should also be interoperability between system components so that components from different vendors can work together to provide the necessary functionality. System components should also exhibit interoperability so that a system component from one vendor can replace that of another without any change in functionality.

In the past, ITS field devices did not exhibit compatibility to any significant degree. Most manufacturers used their own proprietary protocols to communicate, which made sharing of a communications link by multiple manufacturers impossible. Management applications had to develop drivers for each brand and type of device. Some projects did achieve a sense of compatibility by mandating support for a system integrator's protocol such as Protocol 90 or Management Information System for Traffic (MIST). These protocols, however, are for use with traffic signal controllers and not other ITS field devices. At the communications infrastructure level, NTCIP protocols are device independent.

To achieve interoperability, the effort to standardize communications involved three standards organizations: the American Association of State Highway and Transportation Officials (AASHTO), the Institute of Transportation Engineers (ITE), and the National Electrical Manufacturers Association (NEMA). These three organizations brought together public sector

representatives, consultants, and equipment manufacturers to work out a common language for expressing information and sets of messages to exchange the information. By agreeing upon these common protocols, management applications can send information to a type of field device and have it understood by the device, as well as, receive information from the device and understand what it is saying.

By defining the meaning of the words in the common language and their effect upon a device, the NTCIP standards achieve interchangeability. The standards establish a minimum level of common functionally. While a system and its components are free to go beyond the common functionality, subscribing to the NTCIP standards ensures that a level of interchangeability is always present.

For TxDOT to subscribe to NTCIP means that its specifications must specify requirements for NTCIP standards. So far, TxDOT has a start in that direction in a number of areas. After reviewing ITS field device specifications that appear on the TxDOT Expressway webpage, the researcher found that the statewide use specifications for traffic signal controllers and dynamic message signs have fully detailed requirements (4). Similar specifications for video imaging vehicle detection systems (VIVDS) and spread spectrum radios also cite NTCIP requirements, but only in general, with no details of what that entails. Other devices such as onstreet masters, closed circuit television, electrical lighting and management systems, and ramp metering controller do not have any NTCIP requirements. This research further discusses that transition by looking at the documentation to specify NTCIP requirements and then developing test procedures to test for compliance to the NTCIP requirements.

CCTV DOCUMENTATION

This section looks at the documentation to add NTCIP requirements to TxDOT's CCTV specifications and test procedures to verify those requirements. Special Specification 6025 - CCTV Field Equipment does not currently address NTCIP requirements (1). Suggested modifications to that specification appear in Appendix A. The first part of this section presents the reasoning behind those changes.

Before one can properly address specific NTCIP requirements, one needs to correlate the general requirements (in SS 6025) to the object definitions and functionality in NTCIP standards (1). The recommended method of ensuring that test procedures meet user needs and requirements

is through the use of an RTM. Text and examples illustrate the composition and elements of an RTM. The text elaborates on the steps and process of creating an RTM. Table 3 then presents a fully developed RTM for SS 6025 (1). While it simplifies the expression of user needs, the RTM does map them to requirements and references in SS 6025, cites the relevant NTCIP objects, and correlates these to specific test procedure identifiers. The test procedure identifiers relate to a set of CCTV test procedures that appear in Appendix C.

With the NTCIP object definitions and functionality identified, one can proceed to develop test cases and test procedures to ensure that a device implements the definitions and functionality correctly. In some cases, this may be a simple process of reviewing test procedures that are already in the public domain. In other cases, one may need to consider a formal test plan.

The first case applies to CCTV. A number of test procedures and test cases already exist. A review of the procedures and test cases indicates where they apply. Assigning them test case numbers and references allows a mapping to the RTM. In the case of the documentation for traffic signal controllers, the process was the development of internal documentation that sets out a plan, identifies test cases, and outlines individual test cases.

The last part of the testing documentation to consider is the presentation of the testing results. While most testing tools produce some type of test report, the researcher believes that such a report should include how a device was tested. The researcher presents an approach that uses documentation that may appear in the NTCIP standards.

Special Specification CCTV Field Equipment

The current special TxDOT specification for CCTV equipment does not address NTCIP. Appendix A is a modified version of the SS 6025 (1). The text written in italics or shown crossed out illustrates modifications that come from other special specifications that include a reference to NTCIP requirements.

The first modification is an additional sentence in the description clause, which is a reference to another special specification dealing with the specifics of NTCIP requirements. The typical wording that appears in other special specifications is:

The following special specifications is referenced in this specification: "National Transportation Communications for ITS Protocol for CCTV Equipment"

The next modification deals with the clause describing the communications interface. The exact wording may differ in particular special specifications. For SS 6025, however, the chances consist of referencing the Special Specifications NTCIP for CCTV Equipment, removal of optional data rates that are in conflict with NTCIP, and removal of any functional requirements related to the protocol used (*1*).

Another change is the reference to "EIA-232 C/D port." The term "RS (Recommended Standard)-232 Serial port" generalizes the requirement so that specifics come from Special Specifications NTCIP for CCTV Equipment. The change uses the term "RS" in place of "EIA" because the formal definition of the 9-pin version of the RS-232 interface comes from EIA (Electronics Industry Alliance)-574. The change drops the term C/D because the NTCIP standards call for a newer version of EIA-232 standard (Version F).

Initial Requirements Traceability Matrix

Once there is a specification that defines the functional requirements, the next step is to develop an RTM that maps the requirements to NTCIP object definitions. The purpose of the RTM is to help ensure the functionality of the requirements maps to one or more NTCIP objects that either define, control, or provide status related to that function. This ensures that NTCIP conforms to the requirements and that requirements have NTCIP support.

A traceability matrix verifies that all stated and derived requirements are allocated to at least one NTCIP object (forward tracing). The matrix lists the source of requirements (backward tracing). While the RTM in this document focuses on NTCIP, a matrix can include tracing to things other than NTCIP objects such as capabilities, physical requirements, test procedures, etc. For example, a user need could be to operate outdoors and require equipment over a certain temperature range.

 Table 1 is the start of an RTM that references NTCIP objects. Table 3 is a fully expanded

 RTM that adds references to test procedures.

The procedure for creating an RTM consists of:

- identifying user needs (nouns) that correspond to the names of functions,
- referencing the text that describes the action (verb) or usage of the user need,
- providing a cross-reference to an item or clause of the requirements document, and
- correlating the user need to the NTCIP standard or object that supports the need.

SS 6025 Closed Circuit Television NTCIP Requirements									
Traceability Matrix									
User Need	Requirement	TxDOT	NTCIP Object Support						
		SS 6025							
Keference									
	Traceable to N	TCIP 1205-C							
Provide Remote	e Control	F							
Shutter	Provide remotely	2.B.1	None						
Speed	selectable shutter speed								
Zoom	Provide a lens with	2.B.2	3.2.8 rangeZoomLimit						
	capabilities for remote		3.3.3 timeoutZoom						
	control of zoom operations		3.5.3 positionZoomLens						
			3.5.8 positionQueryZoom						
Long-Term	Provide control receivers	6.	None						
Exposure	for Digital Signal								
	Processing (DSP) of long-								
	term exposure control								
Auto-Focus	Provide control receivers	6.	3.2.9 rangeFocusLimit						
	for Digital Signal		3.3.4 timeoutFocus						
	Processing of auto-focus		3.6.4 systemLensFeatureControl						
	control		3.6.5 systemLensFeatureStatus						
			3.6.6 systemLensEquipped						
Auto/Manual	Provide units with control	6.	3.2.9 rangeFocusLimit						
Focus	receivers for DSP		3.3.4 timeoutFocus						
Control	Function – auto/manual		3.6.4 systemLensFeatureControl						
	focus control		3.6.5 systemLensFeatureStatus						
			3.6.6 systemLensEquipped						

Table 1. Requirements Traceability Matrix with NTCIP Objects.

In the case of conformance and compliance related to NTCIP, the list of user needs is limited to parameters, controls, and status information that relate to communications. For CCTV, the needs primarily come from remote control. Any quantities, limits, or values (constraints) that apply to the needs should also appear in the list. Examples of constraints are number of presets, tilt limits, and label color choices. The next column in the RTM is for stating the requirement with some action. This typically consists of a short sentence putting the user need in context. For example, if the need is a zoom capability, then stating that zoom has to be controlled remotely puts it in the context of NTCIP. The third column lists the reference to the item, clause, or heading number in the specification from which user need and requirement is derived. This provides the backward traceability. To complete the RTM for this stage, one enters the object names from NTCIP Standards that relate to user and requirement. The NTCIP object names usually include one or more of the same words used to define the user needs. The NTCIP object description field may also contain a reference. It is typical to cite the same NTCIP objects in many places. At this point, the RTM should look similar to Table 1.

A number of NTCIP standards now include RTM information. Table 2 is part of an RTM that appears in NTCIP 1203 – Object Definitions for Dynamic Message Signs (DMS) (NTCIP 1203-DMS) (5). While the RTM in that standard includes additional columns of information, the functional requirement and object names that support the requirement are the essential items. Even if an NTCIP standard includes an RTM, it is advisable to generate one based upon the requirements defined in TxDOT specifications. The need to do this becomes apparent when one realizes that the NTCIP standards may not provide support for all functional requirements in a specification. NTCIP standards also may require some functionality that does not correspond to a requirement in the specification. For example, the functional requirement to provide remotely selectable shutter speed or control of long-term exposure in SS 6025 does not have object support in NTCIP 1205–CCTV) (1,6). NTCIP 1205-CCTV has support for remote on-screen menu control that does not appear in SS 6025 (6,1).

FR ID	Functional Requirement	Interface ID	Interface	Dialog ID	Object ID	Object
D.3.1.1	Determine Device Component Information		D.4.2.1			
		D.4.3.4.1	Module Table			
					D.2.2	globalMaxModules
		D.4.3.4.2	Module			
					D.2.3.1	moduleNumber
					D.2.3.2	moduleDeviceNode
					D.2.3.3	moduleMake
					D.2.3.4	moduleModel
					D.2.3.5	moduleVersion
					D.2.3.6	moduleType

Table 2. Requirements Traceability Matrix from NTCIP 1203-DMS.

After filling out the NTCIP object support column (see Table 1), the list of object names will identify specific objects and conformance groups that an implementation needs to support and suggest what a potential test procedure needs to address. The list will also indirectly identify objects and requirements that may not be in the requirement specification. After generating an RTM based solely upon the user needs and requirements in SS 6025, the following is the list of objects or groups in NTCIP 1205-CCTV v01.10 that are traceable to a requirement defined in SS 6025 (1,6):

- 3.2.1 rangeMaximumPreset
- 3.2.2 rangePanLeftLimit
- 3.2.3 rangePanRangeLimit
- 3.2.4 rangePanHomePosition
- 3.2.6 rangeTiltUpLimit
- 3.2.7 rangeTiltDownLimit
- 3.2.8 rangeZoomLimit
- 3.2.9 rangeFocusLimit
- 3.2.10 rangeIrisLimit
- 3.2.11 rangeMinimumPanStepAngle
- 3.2.12 rangeMinimumTiltStepAngle

- 3.3.1 timeoutPan
- 3.3.2 timeoutTilt
- 3.3.3 timeoutZoom
- 3.3.4 timeoutFocus
- 3.3.5 timeoutIris
- 3.4.1 presetGotoPosition
- 3.4.2 presetStorePosition
- 3.4.3 presetPositionQuery
- 3.5.1 positionPan
- 3.5.2 positionTilt
- 3.5.3 positionZoomLens

- 3.5.4 positionFocusLens
- 3.5.5 positionIrisLens
- 3.5.6 positionQueryPan
- 3.5.7 positionQueryTilt
- 3.5.8 positionQueryZoom
- 3.5.9 positionQueryFocus
- 3.5.10 positionQueryIris
- 3.6.1 systemCameraFeatureControl

- 3.6.2 systemCameraFeatureStatus
- 3.6.3 systemCameraEquipped
- 3.6.4 systemLensFeatureControl
- 3.6.5 systemLensFeatureStatus
- 3.6.6 systemLensEquipped
- 3.7 CCTV Alarm Objects
- 3.9 CCTV Discrete Output Objects
- 3.11 CCTV Label Objects

Comparing the above list to the heading numbers and object names in NTCIP 1205-CCTV, the following is a list of object names and groups in NTCIP 1205-CCTV v01.10 that do not trace to a requirement defined in SS 6025 (6,1):

- 3.2.5 rangeTrueNorthOffset
- 3.8 CCTV Discrete Input Objects
- 3.10 CCTV Zone Objects
- 3.12 CCTV On-Screen Menu Control Objects

At this point, one may want to consider adding a user need or functional requirement in the specification to address the functionality expressed by the objects. If one determines that they are not necessary, the special specification that defines NTCIP for the specific equipment need not require them even if the NTCIP standard indicates that they are mandatory. This gets at the heart of the distinct difference between conformance to the NTCIP and compliance with a TxDOT specification. To show conformance to a NTCIP standard, an implementation needs to support mandatory objects and have tests run to determine if it is correct. To show compliance with TxDOT specifications, an implementation does not need to support objects that have no functional requirement, and one does not need to run tests on those objects.

One can also use an RTM to summarize user needs and requirements that have no support within the NTCIP standard. These non-supported items would show up as a blank in the NTCIP object support column of the RTM. The following requirements in TxDOT Special Specification 6025 do not trace to an object name or group in NTCIP 1205-CCTV v01.10 (1,6):

- Shutter Speed
- Long-Term Exposure
- Remote White Balancing Control
- Auto and Manual White Balance Control

After identifying the unsupported needs and requirements, one may want to consider whether they are truly required. If so, a manufacturer may support them through proprietary objects. The manufacturer may also have a set of procedures that provide functional testing.

The next step is to determine if the TxDOT specification needs to address additional objects. The following is the list of object names or groups referenced in NTCIP 1205 – Object Definitions for Closed Circuit Television (CCTV) Camera Control (NTCIP 1205-CCTV) v01.10 (and defined in NTCIP 1201 – Global Object Definitions [NTCIP 1201-GLO]) that do not trace to a requirement defined in SS 6025 (6,7,1):

- B.7 Global Configuration Conformance Objects
- B.8 Security Conformance Group

NTCIP 1205-CCTV does not address application, transport, and subnetwork-level requirements (6). Even though the object definitions use the Simple Network Management Protocol (SNMP) macro to describe them, there are several choices for encoding information and exchanging it over different media. A discussion of these choices appears in the next section. However, there may be objects and conformance groups that relate to the setup, control, and status at the application, transport, or subnetwork-levels. Special specifications related to NTCIP that include references to specific application, transport, and subnetwork-level requirements may not be valid for statewide use. For example, one project may require an RS-232 interface while another may require an Ethernet interface. The objects and conformance groups would be different.

Based upon the general description of the communications protocols appearing in SS 6025, the assumption is that the application-level protocol is SNMP as defined in NTCIP 2301 – Simple Transportation Management Framework Application Profile (NTCIP 2301-STMF) (1,8). The following is the list of object names or groups referenced in NTCIP 2301:2001 that do not trace to a requirement defined in SS 6025 (1,8):

- A.7.1.1 System Group
- A.7.1.2 SNMP Group
- A.5.4 SNMP Configuration Conformance Group

Since the communications protocol description does not mention a networking capability, the assumption is that the transport-level protocol is the Null protocol as defined in NTCIP 2201 – Transportation Transport Profile (NTCIP 2201-T2). There are no objects or groups called for in this standard (9).

From SS 6025's description, the assumption is that the subnetwork-level protocol is the point-to-multi-point protocol (PMPP) as defined in NTCIP 2101 – Point to Multi-Point Protocol Using RS-232 Subnetwork Profile (NTCIP 2101-PMPP/RS232) (1,10). The following is the list of object names or groups referenced in NTCIP 2101-PMPP/RS232 that do not trace to a requirement defined in SS 6025 (10,1):

- A.7.1 HDLC (High-Level Data Link Control) Group
- A.7.2 RS232 Asynchronous Group
- A.7.3 HDLC Group Address Group

With the base specification put in the form of an RTM one can then address the language that goes into a specific specification dealing with NTCIP related requirements. The completed RTM (less the test procedure column) shown in Table 3 (see page 21) serves as the basis for a specification for NTCIP for CCTV equipment.

Special Specification NTCIP for CCTV Equipment

Since TxDOT does not have a document that addresses NTCIP requirements related to CCTV Equipment, Appendix B is the researcher's suggestion for a TxDOT special specification that deals with the particulars. Using TxDOT 2004 Specifications - Special Specification 6026 – National Transportation Communications for ITS Protocol for Dynamic Message Signs (SS 6026) as a model for format and wording, the document in Appendix B covers the communications protocol and data dictionary (object definitions) requirements that are specific to CCTV equipment. Appendix B serves the purpose of discussion and provides suggested wording (*11*).

In Appendix B, Item 1 - Description is typical boilerplate wording that TxDOT uses in similar documents. Item 2 - Requirements begins with two paragraphs of boilerplate information.

Items 2.A through 2.G deal with specific NTCIP requirements as they relate to CCTV equipment. Items 2.H through 2.J cover TxDOT specific values and ranges, an operational requirement, and documentation requirements. Item 3- Testing and Verification and Item 4 - Measurement and Payment are typical boilerplate information, as well.

As in SS 6026, items 2.A though 2.G of Appendix B cover the communications protocol and profile requirements (*11*). Functional requirements dictate the choices but NTCIP 9001 – The NTCIP Guide (NTCIP Guide) provides a framework for how all the various protocols and profiles fit together (*12*). As stated in the NTCIP Guide,

"To ensure a working system, deployers must specify and/or select an NTCIP protocol or profile at each level."

Item 2.A defines the subnetwork level profile requirements. Since the original SS 6025 references an interface through an EIA-232 C/D port, the NTCIP 2101-PMPP/RS232 standard is now the reference (*1*, *10*). This is the standard that most CCTV manufacturers support. TxDOT should understand that this standard might not be appropriate to all CCTV implementations. Some may warrant a dial-up (NTCIP 2103 – Point-to-Point Protocol over RS-232 Subnetwork Profile [NTCIP 2103-PPP]) or Ethernet (NTCIP 2104 – Ethernet Subnetwork Profile [NTCIP 2104-Ethernet]) subnetwork (*13*,*18*).

Item 2.B defines the transport level profile requirements. Since the original SS 6025 does not mention any networking capability, NTCIP 2201-T2 serves as the reference (9). Most CCTV manufacturers support this standard. NTCIP 2201-T2 addresses a multiplexing capability by defining various parsing and encapsulation methods (9). The multiplexing scheme would permit several different protocols to exist at the application level. Since CCTV devices do not appear to need this functionality, the requirement only references parsing method 1 and encapsulation method 1.

Item 2.C defines the application level profile requirements. The reference is to NTCIP 2301-STMF (8). This standard defines the use of SNMP as the industry standard protocol for encoding data and exchanging the data between a management application and an ITS field device. NTCIP 2301-STMF actually defines three different protocols: SNMP, Simple Transportation Management Protocol (STMP), and Simple Fixed Message Protocol (SFMP) (8).

SNMP offers simplicity and flexibility but adds overhead with its encoding method. STMP offers flexibility and compact encoding but is not simple to set up and use. SFMP offers simplicity and compact encoding but does so by being less flexible as to what kind of information it handles. Given the amount of information in the exchanges and the frequency of the exchanges, Conformance Level 1 of NTCIP 2301-STMF (only the SNMP portion) is the appropriate choice (8). Most CCTV manufacturers support this conformance level.

Items 2.D through 2.H define the information level profile requirements. These items cover:

- basic configuration and control object definitions/data elements of a CCTV,
- two optional objects that relate to the position of the camera,
- two SNMP-related groups that identify the equipment and provide communications troubleshooting information, and
- three PMPP-related configuration and communications troubleshooting information groups.

Item 2.D identifies the mandatory and optional conformance groups with which the CCTV equipment must comply to meet TxDOT specifications. These consist of:

- CCTV Configuration covering limits, timeouts, and labels
- Extended Functions covering features such as power, heater, focus, alarms, and I/O
- Motion Control covering presets and positioning controls
- Configuration covering manufacturer make and model information

The following is the list of objects names and conformance groups within NTCIP 1205-CCTV that do not correspond to a requirement in SS 6025 and therefore have no reference in Item 2.D (6,1):

- CCTV Discrete Input Objects (clause 3.8 of NTCIP 1205-CCTV)
- CCTV Zone Objects (clause 3.10 of NTCIP 1205-CCTV)
- CCTV On-Screen Menu Control Objects (clause 3.12 of NTCIP 1205-CCTV)

Item 2.E identifies two optional objects from the Motion Control Conformance Group and another object from the Configuration Conformance Group. These three objects do not have a specific requirement in SS 6025 but may be useful (1). The objects positionQueryFocus and positionQueryIris can provide a user with current position information about focus and iris. The globalSetIDParameter indicates any changes have been made to parameter type objects (objects whose value is retained even if power is lost).

Item 2.F identifies three conformance groups that come from NTCIP 2301-STMF (8). The first group is the System Conformance Group. This group is a set of generic object definitions from Request for Comment (RFC) 1213 - Management Information Base for Network Management of TCP/IP-based internets: MIB-II (RFC 1213) that apply to any device implementing SNMP (14). They provide system identification, contact information, and other general information about the equipment. The second group is SNMP Statistics. This group is another set from RFC 1213 that provides counters that keep track of the number of messages received, how many different types of messages there are, the number of errors, how many messages had the wrong community name, and other counters related to SNMP operations (14). The last group is a single object in the SNMP Configuration Conformance Group. The object, snmp-maxPacketSize, defines how many bytes long a message may be at the application level.

Item 2.G defines three conformance groups that are mandatory requirements for devices that are conformant to NTCIP 2101-PMPP/RS232 and support SNMP. The first group is the HDLC Conformance Group (*10*). This group consists of two timers that control response timing, two control parameters that define how many bytes long a message may be at the subnetwork-level, and an identifier of the port number to which the parameters apply. The second group is the RS232 Asynchronous Group. This group defines the port type and data rate at which the interface operates. The group also has counters that keep track of any framing and overrun errors that may occur. The last group is HDLC Group Address. This group defines a number of addresses that a device would listen to for broadcast messages. For example, setting the time-of-day is one example that might be a broadcast message.

Items 2.F and 2.G are information-level object definitions that are mandatory to support if NTCIP 2301-STMF and NTCIP 2101-PMPP/RS232 are applicable (*8,10*). They are mandatory if a CCTV Equipment is conformant to the NTCIP standards. For compliance to TxDOT specifications, the researcher does not believe that HDLC Group Address Conformance Group is a requirement.

TxDOT should consider whether all or parts of the SNMP Statistics, SNMP Configuration, and HDLC Conformance Group require support. The researcher believes that SNMP statistics would be helpful in analyzing the amount and type of messages and useful in

troubleshooting communications problems. However, their utility in a non-networked environment is questionable. Entries in the Minimum Project Requirements table could replace snmp-maxPacketSize in SNMP Configuration and several objects in the HDLC Group. From a system's perspective, if other types of devices use the same application and subnetwork level profiles, the object support should be the same across all of them. For CCTV equipment, however, they are not essential.

Items 2.A through 2.G cover the four profile/protocol levels. To ensure that a specification identifies and addresses each level, the researcher recommends that item numbers specifically identify each level. For example, adding the following item numbers would clearly indicate and identify the requirements at each level:

- 2.1 Subnetwork-level requirements
- 2.2 Transport-level requirements
- 2.3 Application-level requirements
- 2.4 Information-level requirements
- 2.4 Project requirements and documentation

Item 2.H covers Minimum Project Requirements. The intent of NTCIP is to provide communication and information standards that are applicable to the transportation industry as a whole. The SYNTAX values that appear in numerous objects are more for purposes of understanding the encoding size rather than any state or project specific values. The technique to add state or project specific variances is to subrange the values. The typical means of expressing the variances is through Minimum Project Requirements. The following is a discussion of the value of each object in the table.

The minimum value of labelMaximum is 16. The object represents the minimum number of labels that the CCTV equipment must support. SS 6025 does not define a value so this is an arbitrary choice (*1*). One manufacturer supports 64.

The value for labelColor is white. NTCIP 1205-CCTV specifies this as the default color that is mandatory to support. SS 6025 indicates that text should be white letters and black outline (6,1).

The value of rangeMaximumPreset is 16. The object defines the minimum number of preset positions that CCTV equipment must support. Presets include pan, tilt, zoom, and focus
values. SS 6025 does not define a value so this is an arbitrary choice (1). One manufacturer supports 64.

The values for rangePanLeftLimit and rangePanRightLimit are both set to 35999. The objects define the maximum angle that a camera can pan from the home position. SS 6025 specifies a horizontal movement of 360° full, contiguous rotation movement (*1*). If the requirement is to have the CCTV continuously to the right or left without any limit, then this value should be 65535.

The value of rangePanHomePosition is set to 0. The object defines an arbitrary point on a circle from which to measure the left and right limits. The value of rangeTrueNorthOffset is set to 65535. This means that there is no support for a true North offset from the home position. SS 6025 does not specify equivalent requirements (1).

The researcher has set the minimum requirement for rangeTiltUpLimit and rangeTiltDownLimit to +4000 and -9000, respectively. These values correspond to the values in SS 6025, 40 degrees up and 90 degrees down (*1*). In looking at several special specifications for district use, the requirements ranged from 20 to 90 degrees up and from 90 to 110 degrees down.

The researcher has set the minimum requirement for rangeZoomLimit to 65535. This corresponds to a scalar focus positioning value of telephoto. The values for rangeFocusLimit and rangeIrisLimit are set to 0. A 0 value corresponds to non-support of the limits. The values for rangeMimimumPanStepAngle and rangeMimimumTiltStepAngle are set at 10. This corresponds to value of 0.1 degrees. SS 6025 does not define equivalent requirements (*1*). The value is somewhat arbitrary but is one manufacturer's supported value.

The value of systemCameraEquipped is set to 128 to indicate that the CCTV is only to have controllable camera power. NTCIP 1205-CCTV allows for a controllable heater, wiper, washer, and blower (6). However, SS 6025 does not list these as requirements and, therefore, the value represents the required support (1).

The value of systemLensEquipped is set to 172 to indicate that the CCTV is to have controllable Auto Iris and Auto Focus. SS 6025 defines the Auto Iris requirement in Item 2 and the Auto Focus requirement in Item 6 (I).

The values for zoneMaximum, zoneCameraEquipped, and menuControl are set to N/A (not applicable). SS 6025 does not require zones or on-screen menu control features (*1*).

19

The definition of the last two items in the Minimum Project Requirements table comes from NTCIP 1201-GLO (7). The communityNamesMax value is set to 3. This corresponds to an administrator having the ability to define three different community names (users) that could allow different access rights to the information in a CCTV. For example, one user could have full read-write access while another could be limited to just read-only access. It is manufacturer specific how communityNameAccessMask would be set to do this.

Returning to discussion of the numbered items of Item 2, Item 2.I – Hardware Limitations relates a general functional requirement of how SNMP operates. Since NTCIP 2301-STMF covers this, the researcher does not feel its inclusion is necessary (8).

Item 2.J – Documentation discusses the requirements for supplying Management Information Bases (MIBs). In the first year's report on this project, the researcher recommended that TxDOT create or define MIBs that reflect TxDOT specifications. This MIB should be the one that a management application uses and the one used to perform any testing. One reason for doing this is that changes in versions of any standards can introduce incompatibilities. For example, NTCIP 1201-GLO Version 2 inadvertently made a change that introduced an incompatibility with implementations built to Version 1 (7). If the testing of implementations of Version 1 used the MIB defined in Version 1, there was no problem. If the testing of implementations of Version 2 used the MIB defined in Version 2, there was no problem. The problem was uncovered only when the versions were crossed.

Expanded Requirements Traceability Matrix

Once the specification of NTCIP for specific equipment is defined, test procedures that verify that an implementation meets the specification are created and cross-referenced in an RTM. Table 3 is the completed RTM.

	SS 6025 Closed Circuit Television Requirements Traceability Matrix									
User Need	Requirement	TxDOT SS 6025 Reference	NTCIP Object Support	Test Procedure Identifier						
	Traceable to NTCIP 1205-CCTV									
Provide Remote	Provide Remote Control									
Shutter Speed	Provide remotely selectable shutter speed	2.B.1	None	None						
Zoom	Provide a lens with capabilities for remote control of zoom operations	2.B.2	3.2.8 rangeZoomLimit3.3.3 timeoutZoom3.5.3 positionZoomLens3.5.8 positionQueryZoom	Config-TC006 Zoom-TC003 Zoom-TC001 Zoom-TC002						
Long-Term Exposure	Provide control receivers for Digital Signal Processing of long-term exposure control	2.B.6	None	None						
Auto-Focus	Provide control receivers for Digital Signal Processing function of auto-focus control	2.B.6	 3.2.9 rangeFocusLimit 3.3.4 timeoutFocus 3.6.4 systemLensFeatureControl 3.6.5 systemLensFeatureStatus 3.6.6 systemLensEquipped 	Config-TC007 Focus-TC003 Lens-TC002 and Lens-TC003 Lens-TC002 and Lens-TC003 Lens-TC001, Lens-TC002, and Lens-TC003						
Auto/Manual Focus Control	Provide units with control receivers for DSP function of auto/manual focus control	2.B.6	3.6.4 systemLensFeatureControl3.6.5 systemLensFeatureStatus3.6.6 systemLensEquipped	Lens-TC002 and Lens-TC003 Lens-TC002 and Lens-TC003 Lens-TC001, Lens-TC002, and Lens-TC003						
I.D. Generator Operation	Provide units with control receivers for DSP function of I.D. generator operation	2.B.6	3.11 CCTV Label Objects	Label-TC001 and Label- TC002						

Table 3	. Regi	uirements	s Tracea	ability	Matrix	with	Test	Proced	ures	(contin	ued).
										(

	SS 6025 Closed Circuit Television Requirements Traceability Matrix						
User Need	Requirement	TxDOT SS 6025 Reference	NTCIP Object Support	Test Procedure Identifier			
Alarm Function Control	Provide units with control receivers for DSP function of alarm function control	2.B.6	3.7 CCTV Alarm Objects	Alarm-TC001, Alarm-TC002, Alarm-TC003, Alarm-TC004, Alarm-TC005, Alarm-TC006, and Alarm-TC007,			
Pan/Tilt Position preset	Provide units with control receivers for DSP function of pan/tilt position preset	2.B.6	3.4.1 presetGotoPosition 3.4.2 presetStorePosition 3.4.3 presetPositionQuery	Zone-TC001 Zone-TC001 Zone-TC001			
Pan Left	Provide units with control receivers for DSP function of pan left	2.B.6	 3.2.2 rangePanLeftLimit 3.2.4 rangePanHomePosition 3.2.11 rangeMinimumPanStepAngle 3.3.1 timeoutPan 3.5.1 positionPan 	Config-TC003 Config-TC011 and Pan-TC002 Config-TC009 Pan-TC003 Pan-TC001, Pan-TC002, Pan-TC003, and Pan-TC004 Pan-TC002			

	SS 6025 Closed Circuit Television Requirements Traceability Matrix						
User Need	Requirement	TxDOT SS 6025 Reference	NTCIP Object Support	Test Procedure Identifier			
Pan Right	Provide units with control receivers for DSP function of pan right	2.B.6	 3.2.3 rangePanRightLimit 3.2.4 rangePanHomePosition 3.2.11 rangeMinimumPanStepAngle 3.3.1 timeoutPan 3.5.1 positionPan 	Config-TC003 Config-TC011 and Pan-TC002 Config-TC009 Pan-TC003 Pan-TC001, Pan-TC002, Pan-TC003, and Pan-TC004 Pan-TC002			
Tilt Up	Provide units with control receivers for DSP function of tilt up	2.B.6	 3.2.6 rangeTiltUpLimit 3.2.12 rangeMinimumTiltStepAngle 3.3.2 timeoutTilt 3.5.2 positionTilt 3.5.7 positionQueryTilt 	Config-TC005 Config-TC009 Tilt-TC003 Tilt-TC001, Tilt-TC002, Tilt-TC003, and Tilt-TC004 Tilt-TC002			
Tilt Down	Provide units with control receivers for DSP function of tilt down	2.B.6	 3.2.7 rangeTiltDownLimit 3.2.12 rangeMinimumTiltStepAngle 3.3.2 timeoutTilt 3.5.2 positionTilt 3.5.7 positionQueryTilt 	Config-TC005 Config-TC009 Tilt-TC003 Tilt-TC001, Tilt-TC002, Tilt-TC003, and Tilt-TC004 Tilt-TC002			

	SS 6025 Closed Circuit Television Requirements Traceability Matrix						
User Need	Requirement	TxDOT SS 6025 Reference	NTCIP Object Support	Test Procedure Identifier			
Zoom and Focus	Provide units with control	2.B.6	3.4.1 presetGotoPosition	Zone-TC001			
Position Preset	receivers for DSP function of		3.4.2 presetStorePosition	Zone-TC001			
	zoom and focus position preset		3.4.3 presetPositionQuery	Zone-TC001			
Zoom In	Provide units with control	2.B.6	3.2.8 rangeZoomLimit	Config-TC006			
	receivers for DSP function of		3.3.3 timeoutZoom	Zoom-TC0003			
	zoom in		3.5.3 positionZoomLens	Zoom-TC001,			
				Zoom-TC002,			
				Zoom-TC003, and			
				Zoom-TC004			
			3.5.8 positionQueryZoom	Zoom-TC002			
Zoom Out	Provide units with control	2.B.6	3.2.8 rangeZoomLimit	Config-TC006			
	receivers for DSP function of		3.3.3 timeoutZoom	Zoom-TC003			
	zoom out		3.5.3 positionZoomLens	Zoom-TC001,			
				Zoom-TC002,			
				Zoom-TC003, and			
				Zoom-TC004			
			3.5.8 positionQueryZoom	Zoom-TC002			
Focus Near	Provide units with control	2.B.6	3.5.4 positionFocusLens	Focus-TC001,			
	receivers for DSP function of			Focus-TC002,			
	focus near			Focus-TC003, and			
				Focus-TC004			
			3.5.9 positionQueryFocus	Focus-TC002			
Focus Far	Provide units with control	2.B.6	3.5.4 positionFocusLens	Focus-TC001,			
	receivers for DSP function of			Focus-TC002,			
	focus far			Focus-TC003, and			
				Focus-TC004			
			3.5.9 positionQueryFocus	Focus-TC002			

	SS 6025 Closed Circuit Television Requirements Traceability Matrix					
User Need	Requirement	TxDOT SS 6025 Reference	NTCIP Object Support	Test Procedure Identifier		
Manual and Auto Iris Control	Provide units with control receivers for DSP function of manual and auto iris control	2.B.6	3.6.4 systemLensFeatureControl3.6.5 systemLensFeatureStatus3.6.6 systemLensEquipped	Lens-TC002 and Lens-TC003 Lens-TC002 and Lens-TC003 Lens-TC001, Lens-TC002, and Lens-TC003		
Iris Open	Provide units with control receivers for DSP function of iris open	2.B.6	3.2.10 rangeIrisLimit3.3.5 timeoutIris3.5.5 positionIrisLens3.5.10 positionQueryIris	Config-TC008 Iris-TC003 Iris-TC001, Iris-TC002, Iris-TC003, and Iris-TC004 Iris-TC002		
Iris Close	Provide units with control receivers for DSP function of iris close	2.B.6	 3.2.10 rangeIrisLimit 3.3.5 timeoutIris 3.5.5 positionIrisLens 3.5.10 positionQueryIris 	Config-TC008 Iris-TC003 Iris-TC001 Iris-TC002		
Camera Power (Latching)	Provide units with control receivers for DSP function of camera power (latching)	2.B.6	3.6.1 systemCameraFeatureControl3.6.2 systemCameraFeatureStatus3.6.3 systemCameraEquipped	Features-TC002 Features-TC002 Features-TC001		
Remote White Balancing Control	Provide units with control receivers for DSP function of white balancing control	2.B.6	None			
Auto and Manual White Balance Control	Provide units with control receivers for DSP function of auto and manual white balance control	2.B.6	None			

SS 6025 Closed Circuit Television Requirements Traceability Matrix						
User Need	Requirement	TxDOT SS 6025	NTCIP Object Support	Test Procedure Identifier		
		Reference				
Auxiliary Output	Provide units with control receivers for DSP function of one auxiliary output	2.B.6	3.9 CCTV Discrete Output Objects	Discrete-TC002		
Quantities, Limit	ts, and Values	•				
Label Color = White with Black Outline	Use Digital Signal Processing (DSP) for built-in I.D. Generator, with white letters and black outline	B.1	3.11.2.5 labelColor	Label-TC001 and PRL- TC001		
Tilt Angle = $+40^{\circ}$ to -90°	Provide a unit with vertical movement of $+40^{\circ}$ to -90°	B.4	3.2.6 rangeTiltUpLimit3.2.7 rangeTiltDownLimit	Config-TC005 and PRL- TC001 Config-TC005 and PRL- TC001		
Tilt Speed = 20° per sec	Tilt speed must be 20° per sec	B.4	3.5.2 positionTilt	Tilt-TC001, Tilt-TC002, Tilt-TC003, and PRL-TC001		
Pan Angle = 360	Provide a unit with horizontal movement of 360° full, contiguous rotation movement	B.4	3.2.2 rangePanLeftLimit3.2.3 rangePanRightLimit	Config-TC003 and PRL- TC001 Config-TC003 and PRL- TC001		
Pan Speed = 100° per sec	The pan speed must be up to 100° per sec.	B.4	3.5.1 positionPan	Pan-TC001 and PRL-TC001		
Pan/Tilt Position Presets = 1	Provide units with remote control Pan/Tilt Position preset	B.6	3.2.1 rangeMaximumPreset	Config-TC002 and PRL- TC001		

	SS 6025 Closed Circuit Television Requirements Traceability Matrix						
User Need	Requirement	TxDOT SS 6025 Reference	NTCIP Object Support	Test Procedure Identifier			
Zoom and Focus Presets = 1	Provide units with remote control of Zoom and Focus position preset	B.6	3.2.1 rangeMaximumPreset	Config-TC002 and PRL- TC001			
Auxiliary Outputs = 1	Provide units with remote control of one auxiliary output	B.6	3.9 CCTV Discrete Output Objects	Discrete-TC002 and PRL- TC001			
Labels	Provide the built-in I.D. Generator that inserts camera I.D.	B.7	3.11.1 labelMaximum	Label-TC001 and PRL- TC001			
Non-identified needs but covered in NTCIP 1205-CCTV							
Not Identified	Provide a true north reference	None	3.2.5 rangeTrueNorthOffset	Config-TC004			
Not Identified	Provide units with remote control of one auxiliary output	None	3.8 CCTV Discrete Output Objects	Discrete-TC002			
Not Identified	Provide units with remote control of heater	None	3.6.1 systemCameraFeatureControl 3.6.2 systemCameraFeatureStatus 3.6.3 systemCameraEquipped	Feature-TC003			
Not Identified	Provide units with remote control of wiper	None	3.6.1 systemCameraFeatureControl 3.6.2 systemCameraFeatureStatus 3.6.3 systemCameraEquipped	Feature-TC004			
Not Identified	Provide units with remote control of washer	None	3.6.1 systemCameraFeatureControl 3.6.2 systemCameraFeatureStatus 3.6.3 systemCameraEquipped	Feature-TC005			
Not Identified	Provide units with remote control of blower	None	3.6.1 systemCameraFeatureControl 3.6.2 systemCameraFeatureStatus 3.6.3 systemCameraEquipped	Feature-TC006			
Not Identified	Provide units with remote	None	3.12.1 menuActivate	Menu-TC001			
	menung		5.12.2 menuControl	Menu-1C001			

	SS 6025 Closed Circuit Television Requirements Traceability Matrix						
User Need	Requirement	TxDOT SS 6025 Reference	NTCIP Object Support	Test Procedure Identifier			
Not Identified	Provide units with remote configuration, control, and labeling of zones	None	3.10.3 zoneCameraEquipped3.10.1 zoneMaximum3.10.2 zoneTable3.4.1 presetGotoPosition	Config-TC010 Zone-TC002 Zone-TC002 Zone-TC001 and Zone- TC003			
			3.4.2 presetStorePosition 3.4.3 presetPositionQuery	Zone-TC001 Zone-TC001			
Traceable to NTCIP 1201-GLO							
Configuration			-	-			
Not Identified	Provide input and display of manufacturer information	None	2.2.2 globalMaxModules	Config-TC001 and GlobalConfig-TC001			
			2.2.3 globalModuleTable [2.2.3.1 moduleNumber 2.2.3.2 moduleDeviceNode 2.2.3.2 moduleMake 2.2.3.2 moduleModel 2.2.3.2 moduleWorsion 2.2.3.2 moduleVersion 2.2.3.2 moduleType]	Config-TC001 and GlobalConfig-TC001			
Not Identified	Provide indication of configuration change	None	2.2.1 globalSetIDParmeter	GlobalConfig-TC002			
Security							
Not Identified	Provide administrative access to all data and user password definitions	None	2.7.1 communityNameAdmin	Security-TC001			
Not Identified	Provide user access passwords	None	2.7.2 maxCommunityNames 2.7.3 communityNameTable	Security-TC002 Security-TC002			

	SS 6025 Closed Circuit Television Requirements Traceability Matrix					
User Need	Requirement	TxDOT SS 6025 Reference	NTCIP Object Support	Test Procedure Identifier		
]	Fraceable to	NTCIP 2301-STMF			
Standardized Communications Protocol	Provide communications signals, data exchange protocol and timing that is compatible with the communications equipment.	B.6	2.7.1 communityNameAdmin2.7.2 maxCommunityNames2.7.3 communityNameTable	Security-TC001 Security-TC002 Security-TC002 <i>Note:</i> Numerous SNMP related functionality test procedures would be applicable to CCTV.		
Not Identified	Provide generic ID of equipment and manufacturer	None	A.5.2 System Conformance Group	SNMP-TC041		
Not Identified	Provide information related to application-level protocol troubleshooting	None	A.5.3 SNMP Statistics Conformance Group	SNMP-TC011		
Not Identified	Accept a data exchange of at least 484 bytes	None	A.5.4 SNMP Configuration Group	SNMP-TC005		
		Traceable	to NTCIP 2201-T2			
Standardized Communications Protocol	Provide communications signals, data exchange protocol and timing that is compatible with non- networked environment	None	Basic functionality of NULL does not require object support	<i>Note:</i> There are several NULL protocol-related test procedures. However, these do not appear to be applicable to CCTV.		
Not Identified	Provide for single application- level protocol	None	Basic functionality of NULL does not require object support			

	SS 6025 Closed Circuit Television Requirements Traceability Matrix						
User Need	Requirement	TxDOT SS 6025 Reference	NTCIP Object Support	Test Procedure Identifier			
	Tra	ceable to NT	CIP 2101-PMPP/RS232				
Standardized Communications Protocol	Provide communications signals, data exchange protocol, and timing that is compatible with RS-232 Serial Port	None	A.7.1 lapBAdminTable A.7.1 lapBOperTable	232-TC025 232-TC025 <i>Note:</i> Numerous PMPP related functionality test procedures would be applicable to CCTV.			
Not Identified	Provide information related to subnetwork-level protocol troubleshooting	None	A.7.2 rs232AsyncPortTable	232-TC024			
Programmable Address	Provide each unit with a unique programmable address	B.6	Basic functionality of PMPP does not require object support	232-TC002			
Group Address	Provide each unit with programmable group addresses	None	A.7.3 HDLC Group Address Conformance Group	232-TC013 and 232-TC014			
Asynchronous	Data must be sent asynchronously		A.7.2 rs232PortTable	232-TC024			
9600 Baud	Use a minimum of 9600 Baud		A.7.2 rs232PortTable	232-TC024			

CCTV Test Procedures

One objective of the research project is to assist TxDOT in developing procedures. Appendix C provides a set of test procedures for testing CCTV pan, tilt, and zoom (PTZ) controllers. The test procedures include a prequalification test case that looks at general object support and supported values and 57 test cases that look at specific functionality expressed by the objects.

The prequalification test case retrieves minimum project requirements and maximum values, checks for whether the device implements the required objects, and performs a sampling of the supported values. The minimum project requirements and maximum values relate to those specified in the special specification on NTCIP for CCTV Equipment (Appendix B). These correspond to such things as the number of labels and the limits of panning, tilting, and zooming. The test case, as written, simply retrieves the value from a device. It does not make any value judgment as to whether the object meets the minimum required value. Although the test procedure could check for a specific minimum value, that interpretation is currently the responsibility of the person reviewing the test results. The object presence or instantiation test uses the information in a local MIB to indicate what objects should be present and then performs a read or SNMP "get" of all possible object instances. Whereas most test procedures use an SNMP "getNext" command to get the next object instantiated or "walk the MIB," the test case uses an SNMP "get" of specific instances. A management application would normally use an SNMP "get" to access information and the test case emulates that operation. The sampling test portion of the test uses an externally defined file of test values, and the test steps use a write or SNMP "set" to store the test value in a device.

The sampling test portion of the test case uses the external file of test values to customize the procedure to check compliance to TxDOT specifications rather than conformance to the NTCIP standard. From a NTCIP perspective, a device may limit the number of instances of an object. For example, the tilt up limit in NTCIP 1205-CCTV is 0 to 360 degrees. A manufacturer can subrange that limit to 10 degrees and still be compliant to the NTCIP standard. The requirement in SS 6025 is 40 degrees (*1*). To test conformance to a TxDOT specification, the test procedure should use a test value of 40. Basing the test case on externally defined values also allows districts to customize the test case to project-specific requirements.

31

The functional test cases use the Enterprise CCTV Test Procedures as their model (*15*). They test whether a device performs the function to which an NTCIP object maps. The organization of the test cases is as follows:

- Alarms
 Label
- Camera Configuration (Camera)
 Lens
- Discrete Input and Output
- Camera Features
- Focus

• Iris

• Global Configuration

- Pan
- Security

Menu

- Tilt
- Zoom
- Some of the functional areas and test cases go beyond the requirements of SS 6025 (*I*). The NTCIP standard addresses additional features that do not have a reference in the special specification. The test cases in Appendix C address some the additional features because some district and one-time use special specifications do reference those features.

The Global Configuration and Security test cases are not specific to CCTV. They would apply to any type of field device that claims compliance to NTCIP. The discussion of the communications and other test procedures appears in a separate section, and a listing appears in Appendix E.

CCTV Test Results

One of the framework recommendations that appears in the first year's report on this project is to use an NTCIP Profile Requirements List as a format for reporting test results. Appendix D is a filled out PRL showing the results of the Prequalification Test Case - TC001 in Appendix C. The PRL from NTCIP 1205:2001 v01.08 Amendment 1 v10 serves as the basic template, but with additional text, that clarifies some revision issues (6).

One of the purposes for including the PRL in NTCIP standards is to use it for selecting appropriate tests to check conformance. A manufacturer and user can also use it as a detailed indication of the capabilities of the implementation. NTCIP PRLs come with specific copyright permission to use them for creating a Protocol Implementation Conformance Specification (PICS). When a PRL includes specific object support and supported values information about an implementation, it becomes a PICS. As the name implies, it is a specification of the features in an implementation. Figure 1 illustrates how the test script changes the object support entries or adds supported values entries to the PRL to produce a PICS test report. Appendix D contains a full PICS test report using the PRL from NTCIP 1205-CCTV (6).

	CCTV Configuration CONFORMANCE GROUP								
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values			
3.2, 3.3 and 3.11	CCTV Configuration Conformance Group		М	YES					
3.2	CCTV Range Objects								
3.2.1	rangeMaximumPreset	S	3.2 : M	YES	0-255	64			
3.2.2	rangePanLeftLimit	S	3.2 : M	YES	0-35999 65535	35999			
3.2.3	rangePanRightLimit	S	3.2 : M	YES	0-35999 65535	35999			
3.2.4	rangePanHomePosition	S	3.2 : M	YES	0-35999 65535	0			
3.2.5	rangeTrueNorthOffset	Ρ	3.2 : M	YES	0-35999 65535	PASSED: 0,1,18000, 35998,359 99,36000, FAILED: - 1,			
3.2.6	rangeTiltUpLimit	S	3.2 : M	YES	0-35999 65535	1500			
3.2.7	rangeTiltDownLimit	S	3.2 : M	YES	0-35999 65535	27000			
3.2.8	rangeZoomLimit	S	3.2 : M	YES	0-65535	65535			

Figure 1. CCTV PRL Information Test Case Results.

The results of the other test procedures and test cases listed in Appendix C would follow the same principle of reusing existing documentation. Figure 2 shows how test case results of the Cabinet Alarm test case would appear as a test result report. Reporting results in this manner not only indicates what was tested and whether it passed or not but also documents the specifics of how it was tested. (A future revision to the test script will add the test script revision number to the report as well.)

Test Case:	Title [.]	Cabinet Alar	m				
Alarm-TC001	Description ⁻	This test case	tests the cal	pinet open alarm and	the label		
	Decemption	associated wi	th it	onior opon alarmane			
	Variables:						
	Pass/Fail	The device ur	erification step				
	Criteria:	included withi	n the Test Ca	ase in order to pass t	he Test Case.		
Test Step	Test Procedure			·	Results		
Number					1.000.000		
1	SET						
	labelText. <alarm< td=""><td>CLabIndex> =</td><td><alarmclab< td=""><td>Text1></td><td>Pass</td></alarmclab<></td></alarm<>	CLabIndex> =	<alarmclab< td=""><td>Text1></td><td>Pass</td></alarmclab<>	Text1>	Pass		
	labelHeight. <ala< td=""><td>rmCLabIndex></td><td>= <alarmcla< td=""><td>abHeight1></td><td></td></alarmcla<></td></ala<>	rmCLabIndex>	= <alarmcla< td=""><td>abHeight1></td><td></td></alarmcla<>	abHeight1>			
	labelColor. <alarr< td=""><td>mCLabIndex> =</td><td>= <alarmcla< td=""><td>bColor1></td><td></td></alarmcla<></td></alarr<>	mCLabIndex> =	= <alarmcla< td=""><td>bColor1></td><td></td></alarmcla<>	bColor1>			
	labelStartRow. <a< td=""><td>alarmCLabInde</td><td>ex> = <alarm@< td=""><td>CLabStartRow1></td><td></td></alarm@<></td></a<>	alarmCLabInde	ex> = <alarm@< td=""><td>CLabStartRow1></td><td></td></alarm@<>	CLabStartRow1>			
	labelStartColumr	n.< <i>alarmCLabl</i>	ndex> =				
	<alarmclabstart(< td=""><td>Column1></td><td></td><td></td><td></td></alarmclabstart(<>	Column1>					
2	SET alarmLabelIn	dex.0 to <alari< td=""><td>mCLabIndex:</td><td>> 00 00 00 00 00</td><td>Pass</td></alari<>	mCLabIndex:	> 00 00 00 00 00	Pass		
	00						
3	USER VERIFY the	at no labels are	e being show	n.	Pass		
4	SET alarmLatchC	lear.0 to 0x00			Pass		
5	Turn on the alarm	and USER VE	RIFY the lab	el for the alarm is	Pass		
	snown.	shown.					
6	GET alarmStatus.	GET alarmStatus.0 and alarmLatchStatus.0					
1					Pass		
	alarmi status = 0x	alarmStatus = 0x80					
Q	SET alarmi atchC						
0	GET alarmStatus	CET alarmStatus 0 and alarmI atabStatus 0					
10					Pass		
10	alarmStatus = 0x	(80			1 455		
	alarmLatchStatus	s = 0x80					
11	USER VERIFY the	e label for the a	alarm is show	n and deactivate	Pass		
	the alarm.						
12	USER VERIFY the	e label for the a	alarm is off.		Pass		
13	GET alarmStatus.	0 and alarmLa	tchStatus.0		Pass		
14	VERIFY Response	e Value			Pass		
	alarmStatus = 0x	(00					
	alarmLatchStatu	s = 0x80					
15	USER VERIFY the	e label for the a	alarm is off.		Pass		
16	SET alarmLatchC	lear.0 to 0x00			Pass		
17	GET alarmStatus.	0 and alarmLa	tchStatus.0		Pass		
18	VERIFY RESPON	ISE VALUE			Pass		
	alarmStatus = 0x00						
	alarmLatchStatu	s = 0x00					
		Test Case	Results	4/00/00			
Tested By:	Jeremy Johnson		Date Tagta d	4/22/06	Pass		
Tastors Nata				407			
Test Case Notes:		nera with I-Cor	101 55N 449	1497			
version History:	v1.0 – Initial Draft	09/20/05	а.IГ. а. а. т. т. т. а. а.	4/00/05			
	V1.1 – Removed deprecated labelFont lype 11/03/05						

Figure 2. CCTV Cabinet Alarm Test Case Results Form.

COMMUNICATIONS LEVEL TEST PROCEDURES

In order to communicate, any specification of an ITS field device needs to address requirements for the communications protocols and physical interfaces. For compliance to NTCIP, this essentially consists of specifying one or more or the application, transport, and subnetwork-level standards and the plant level physical interface. The NTCIP Guide provides information on how to select the various standards from each level and the options to use (*12*).

All of the communications level standards include functional requirements. Some of them also include object conformance groups. If these functional requirements and conformance groups appear in a TxDOT specification, then there are test procedures that would apply. A number of test procedures that relate to communications are already in the public domain. The following sections provide general descriptions of what is available. Appendix E provides further details on individual test cases.

Application Level

There are a number of test procedures that validate the proper operation of the SNMP protocol that apply to an implementation conformant to NTCIP 2301-STMF (8). The functional areas that they address are:

- General SNMP Commands
- Error Responses to Commands
- Community Name Validation
- Statistical Information
- Standard Data Type Encoding
- Opaque Ending

An implementation may incorporate the STMP protocol that NTCIP 2301-STMF specifies (8). A number of test procedures are applicable if that is the case. The test procedures address the Encoding and Decoding of the Data Types

Transport Level

Several test procedures are available for the NTCIP 2201 – Transportation Transport Profile (NTCIP 2201-T2) (9). These procedures address:

- Invalid Protocol Identifier
- Maximum Packet Size
- Support for the ipNetToMedia Conformance Group

Subnetwork Level

At the subnetwork-level, procedures for checking the functionality and objects in two conformance groups may be applicable. The test procedures for PMPP as defined in NTCIP 2101-PMPP/RS232 are:

- Short Address Validation
- Long Address Support and Validation
- Broadcast and Polling
- Group Address Support and Validation
- Polling
- Control Byte
- Invalid Protocol Identifier
- Field Check Sum
- RS232 and HDLC Conformance Groups
- Frame Size and Buffering
- Data Rates and Response Time

TRAFFIC SIGNAL CONTROLLER DOCUMENTATION

TxDOT Specifications for Traffic Signal Controllers

TxDOT DMS 11170-TSC defines the general and the NTCIP specific requirements for traffic signal controllers and their assemblies (2). It covers both the physical, environmental, and functional requirements and the NTCIP requirements. The document devotes about one-tenth of its material to NTCIP compliance. It follows the NTCIP Guide's recommendations on how to

spell out requirements. Besides referencing NTCIP 1202 – Object Definitions for Actuated Signal Controller Units (NTCIP 1202-ASC), DMS 11170-TSC also includes an Object Range Value table as suggested in the NEMA TS 2 Standard (16,2,17).

There are three suggestions for additions to DMS 11170-TSC:

- Reference communications-level conformance groups and object definitions.
- Refer to NTCIP 2103-PPP for dial-up communications (13).
- Refer to NTCIP 2104-Ethernet for Ethernet communications (18).

Several of the communications-level standards contain conformance groups and object definitions that relate to parameters and status information at that level. The standards themselves designate certain groups as being mandatory, but some implementations do not include them. For example, there are object conformance groups related to system information, SNMP statistics, and HDLC that may apply. Sections 2.F and 2.G of Appendix B provide example wording of what to include. Please also keep in mind that different transport and subnetwork-level protocols have different conformance groups.

DMS 11170-TSC references dial-up communications but does not cite any particular standard to follow (2). NTCIP 2103-PPP defines the protocol for use in this type of point-to-point communication (*13*). It also defines an authentication protocol that protects against unauthorized access. The newest version of NTCIP 2103-PPP includes object definitions that standardize the information related to modem initialization and phone numbers (*13*).

Although it is not required by any NTCIP standard, most current state-of-the-art traffic signal controllers include an Ethernet interface. A number of recent projects involving DMSs and CCTVs include a reference to Ethernet (19,20). While there may not be an immediate need to address an Ethernet interface, DMS 11170-TSC could list it as optional (2).

Initial Requirements Traceability Matrix

Developing a full requirements traceability matrix for a traffic signal controller is beyond the scope of this project. This is especially true if the RTM needs to include user needs and provide cross references to a full set of test case identifiers. However, DMS 11170-TSC does have a set of NTCIP requirements (2). This research prototyped some diamond-controller test procedures, and other applicable procedures exist. Given these limitations and available

37

information, the researcher does provide an initial RTM. A first-order RTM also provides some insight into the work effort needed to develop a full set of test procedures.

In Table 4, an RTM enumerates the NTCIP requirements in DMS 11170-TSC, cites the clause or heading defining the requirement, and lists the NTCIP objects that relate to the requirement. In most cases, the test procedure identifier is blank. However, where one of the test cases in Appendix G is applicable, the matrix cross-references it.

Several of the entries in the Test Procedure Identifier column add a prefix to some of the test case identifiers. The prefix is a reference to the device type source. For example, the globalMaxModules object lists (CCTV) Config-TC001 and (ASC) GloCon-TC001, which are equivalent and equally applicable. The column entry lists both because, currently each group or organization working on test procedures for a specific ITS field device is generating its own set of procedures for the objects that appear in NTCIP 1201-GLO (7). The NTCIP working groups are trying to resolve this duplication. However, TxDOT will have a similar situation. Is there a single set of procedures for SNMP, RS-232, and the other communications level standards that can apply to a number of ITS field devices? The researcher believes that there should be a single set of procedures that test for compliance and/or conformance of the common conformance groups and protocols. However, until there is research into whether this is possible, each ITS field device may end up with its own set of test procedures.

	DMS 11170-TSC						
	Re	quirements	s Traceability N	Aatrix			
User	Requirement	TxDOT	NTCIP	Object Support	Test Procedure		
Need		DMS 11170			Identifier ¹		
		Traccable					
Dhaga	Conformance Crown	I raceable	IONICIP 1202-ASC	,			
rnase	Implement all mandatory objects and	NTCIP	maxPhasas		$\mathbf{DPL} \mathbf{TC} 001^2$		
	all mandatory conformance groups	Compliance	nhaseTable		PRI_TC001		
	defined in "Actuated Signal	Compliance	phase rable	phaseNumber	DetOns_TC001		
	Controller Object Definitions,"			phaservalleer	DetOps-TC001 – DetOps-TC018 ³		
	NTCIP 1202:1996: Phase			phaseWalk	· ·		
	Conformance Group			phasePedestrianClear			
				phaseMinimumGreen	DetOps-TC001 -		
					DetOps-TC018		
				phasePassage	DetOps-TC001 -		
					DetOps-TC018		
				phaseMaximum1	DetOps-TC001 -		
					DetOps-TC018		
				phaseMaximum2			
				phaseYellowChange	DetOps-TC001 -		
					DetOps-TC018		
				phaseRedClear	DetOps-TC001 -		
					DetOps-TC018		

¹ Unless otherwise stated, the Test Procedure Identifier refers to those listed in Appendix G. ² The Protocol/Profile Requirements test case also checks for support of all objects listed in the MIB. ³ The Detector Operations test cases do not test the phase intervals but rely on specific values.

	DMS 11170-TSC							
	Requirements Traceability Matrix							
User Need	Requirement	TxDOT DMS 11170 Reference	NTCIP Object Support	Test Procedure Identifier				
	Not Required ⁴		phaseRedRevert					
	Implement all mandatory objects and		phaseAddedInitial					
	all mandatory conformance groups		phaseMaximumInitial					
	defined in "Actuated Signal		phaseTimeBeforeReduction					
	Controller Object Definitions,"							
	NTCIP 1202:1996: Phase							
	Conformance Group							
	Not Required ⁵		phaseCarsBeforeReduction					
	Implement all mandatory objects and		phaseTimeToReduce					
	all mandatory conformance groups							
	defined in "Actuated Signal							
	Controller Object Definitions,"							
	NTCIP 1202:1996: Phase							
	Conformance Group							
	Not Required ⁵		phaseReduceBy					
	Implement all mandatory objects and		phaseMinimumGap					
	all mandatory conformance groups							
	defined in "Actuated Signal							
	Controller Object Definitions,"							
	NTCIP 1202:1996: Phase							
	Conformance Group							
	Implement the following optional		phaseDynamicMaxLimit					

 ⁴ Optional object not required by DMS 11170-TSC.
 ⁵ Optional object not required by DMS 11170-TSC.

	DMS 11170 TSC							
TT	Kequirements Traceability Matrix							
User	Kequirement	TXDOT DMS 11170	NICIP O	bject Support	Test Procedure			
need		DMS 111/0 Deference			Identifier			
	objects as defined in the "Actuated Signal Controller Object Definitions," NTCIP 1202:1996: phaseDynamicMaxLimit and phase DynamicMaxStep.			phaseDynamicMaxStep				
	Implement all mandatory objects and			phaseStartup	DCT-T0001 ⁶			
	all mandatory conformance groups			phaseOptions				
	defined in "Actuated Signal			phaseRing				
	Controller Object Definitions,"			phaseConcurrency				
	NTCIP 1202:1996: Phase		maxPhaseGroups		PRL-TC001			
	Conformance Group		phaseStatusGroup		PRL-TC001			
			Table	phaseStatusGroupNumber				
				phaseStatusGroupReds	IM-TC0001			
				phaseStatusGroupYellows	IM-TC0001			
				phaseStatusGroupGreens	DetOps-TC001 – DetOps-TC018, IM-TC0001			
				phaseStatusGroupDont Walks	IM-TC0001			
				phaseStatusGroupPedClears	IM-TC0001			
				phaseStatusGroupWalks	IM-TC0001			
				phaseStatusGroupVehCalls	IM-TC0001			
				phaseStatusGroupPedCalls	IM-TC0001			

⁶ The dbCreateTransaction test case only looks at the phaseStartup intervals.

	DMS 11170-TSC							
User Need	Requirement	quirements TxDOT DMS 11170 Reference	NTCIP Object Support		Test Procedure Identifier			
				phaseStatusGroupPhaseOns	Seg-TC001 DetOps-TC001 – DetOps-TC018, IM-TC0001			
				phaseStatusGroup PhaseNexts	IM-TC0001			
			phaseStatus	·	PRL-TC001			
			ControlTable	phaseControlGroupNumber				
				phaseControlGroup PhaseOmit				
				phaseControlGroupPedOmit				
				phaseControlGroupHold				
				phaseControlGroup ForceOff				
				phaseControlGroupVehCall	DetOps-TC001 – DetOps-TC018			
				phaseControlGroupPedCall				
Detecte	or							
	Implement all mandatory objects and	NTCIP	maxVehicleDetectors		PRL-TC001			
	all mandatory conformance groups	Compliance	vehicleDetectorTable		PRL-TC001			
	defined in "Actuated Signal			vehicleDetectorNumber				
	Controller Object Definitions,"			vehicleDetectorOptions				
	NTCIP 1202:1996: Detector			vehicleDetectorCallPhase				
	Conformance Group			vehicleDetectorSwitchPhase				
				vehicleDetectorDelay				

[
	DIVIS 111/0-15C							
	Requirements Traceability Matrix							
User	Requirement	TxDOT	NTCIP Ob	ject Support	Test Procedure			
Need		DMS 11170			Identifier			
		Reference						
				vehicleDetectorExtend	DetOps-TC001 -			
					DetOps-TC018			
	Implement the following optional	-		vehicleDetectorQueueLimit				
	objects as defined in the "Actuated							
	Signal Controller Object							
	Definitions," NTCIP 1202:1996							
	Implement all mandatory objects and	-		vehicleDetectorNoActivity				
	all mandatory conformance groups			vehicleDetector				
	defined in "Actuated Signal			MaxPresence				
	Controller Object Definitions,"			vehicleDetector				
	NTCIP 1202:1996: Detector			ErrancCounts				
	Conformance Group							
	Implement the following optional			vehicleDetectorFailTime				
	objects as defined in the "Actuated							
	Signal Controller Object							
	Definitions," NTCIP 1202:1996							
	Implement all mandatory objects and			vehicleDetectorAlarms				
	all mandatory conformance groups							
	defined in "Actuated Signal							
	Controller Object Definitions,"							
	NTCIP 1202:1996: Detector							
	Conformance Group							
	Implement the following optional			vehicleDetectorReported				
	objects as defined in the "Actuated			Alarms				
	Signal Controller Object							

	DMS 11170_TSC						
	Re	anirement	s Traceahility M	atrix			
User Need	Requirement	TxDOT DMS 11170 Reference	NTCIP Object Support		Test Procedure Identifier		
	Definitions," NTCIP 1202:1996						
	Implement all mandatory objects and			vehicleDetectorReset			
	all mandatory conformance groups defined in "Actuated Signal Controller Object Definitions"		maxVehicle Detector StatusGroups		PRL-TC001		
	NTCIP 1202:1996: Detector		vehicleDetector		PRL-TC001		
	Conformance Group		StatusGroupTable	vehicleDetectorStatusGroup Number			
				vehicleDetectorStatusGroup Active			
				vehicleDetectorStatusGroup Alarms			
			maxPedestrian Detectors		PRL-TC001		
			pedestrian		PRL-TC001		
			DetectorTable	pedestrianDetectorNumber			
				pedestrianDetector CallPhase			
				pedestrianDetector NoActivity			
				pedestrianDetector MaxPresence			
				pedestrianDetector ErraticCounts			
				pedestrianDetector Alarms			

	DMS 11170-TSC						
	Re	quirements	s Traceability Ma	trix			
User Need	Requirement	TxDOT DMS 11170 Reference	NTCIP OF	oject Support	Test Procedure Identifier		
Volum	e Occupancy Report						
	Implement all mandatory objects of all optional conformance groups as defined in NTCIP 1202:1996: Volume Occupancy Report	NTCIP Compliance	volume OccupancySequence volume OccupancyPeriod				
	Conformance Group		activeVolume OccupancyDetectors				
			Volume		PRL-TC001		
			OccupancyTable	detectorVolume			
				detectorOccupancy			
Unit	F	1	I		T		
	Implement all mandatory objects of	NTCIP	unitStartUpFlash				
	all optional conformance groups as defined in NTCIP 1202:1996: Unit	Compliance	unitAutoPedestrian Clear				
	Conformance Group		unitBackupTime				
	Implement the following optional objects as defined in the "Actuated Signal Controller Object Definitions," NTCIP 1202:1996		unitRedRevert				
	Implement all mandatory objects of]	unitControlStatus		IM-TC0001		
	all optional conformance groups as		unitFlashStatus				
	defined in NTCIP 1202:1996: Unit		unitAlarmStatus2				
	Conformance Group		unitAlarmStatus1				

	DMS 11170-TSC						
User Need	Requirement	quirements TxDOT DMS 11170 Reference	S Traceability Matrix NTCIP Object Support		Test Procedure Identifier		
			shortAlarmStatus		IM-TC0001		
			unitControl				
	Implement the following optional		maxAlarmGroups				
	objects as defined in the "Actuated		alarmGroupTable				
	Signal Controller Object			alarmGroupNumber			
	Definitions," NTCIP 1202:1996			alarmGroupState			
Special	Function		•	•			
	Implement all mandatory objects of all optional conformance groups as	NTCIP Compliance	maxSpecialFunction Outputs		PRL-TC001		
	defined in NTCIP 1202:1996: Special		specialFunctionOutput		PRL-TC001		
	Function Conformance Group		Table	specialFunctionOutput Number			
				specialFunctionOutput Control			
				specialFunctionOutput Status			
Coordi	nation	1	1				
	Implement all mandatory objects of	NTCIP	coordOperational				
	all optional conformance groups as	Compliance	Mode				
	defined in NTCIP 1202:1996:		coordCorrectionMode				
	Coordination Conformance Group		coordMaximumMode				
			coordForceMode				
			maxPatterns				
			patternTableType				
			patternTable				
				patternNumber			

	DMS 11170-TSC Begyingments Traggability Matrix						
User Need	Requirement	TxDOT DMS 11170 Reference	NTCIP Object Support		Test Procedure Identifier		
				patternCycleTime			
			patternOffsetTime				
			patternSplitNumber				
			patternSequence Number				
			maxSplits		PRL-TC001		
			splitTable		PRL-TC001		
				splitNumber			
				splitPhase			
				splitTime			
				splitMode			
				splitCoordPhase			
			coordPatternStatus		IM-TC0001		
			localFreeStatus				
			coordCycleStatus				
			coordSyncStatus				
			systemPatternControl				
			systemSyncControl				
Time B	ase	NECID					
	Implement all mandatory objects of	NTCIP	timebaseAscPattern				
	all optional conformance groups as	Compliance	Sync				
	defined in NTCIP 1202:1996: Time		max l'imebaseAsc		PKL-TC001		
	Dase Conformance Group		Actions		DDL TC001		
			Table	timebase Asc Action Number	PKL-IC001		
			Table	uncoaseAseAcuonivunibei			

	DMS 11170-TSC Requirements Traceability Matrix							
User Need	Requirement	TxDOT DMS 11170 Reference	NTCIP Object Support		Test Procedure Identifier			
				timebaseAscPattern				
				timebaseAscAuxillary Function				
				timebaseAscSpecial Function				
				timebaseAscActionStatus				
Preem	pt							
	Implement all mandatory objects of	NTCIP	maxPreempts		PRL-TC001			
	all optional conformance groups as	Compliance	PreemptTable		PRL-TC001			
	defined in NTCIP 1202:1996:			preemptNumber				
	Preempt Conformance Group			preemptControl				
				preemptLink				
				preemptDelay				
				preemptMinimumDuration				
	Implement the following optional			preemptMinimumGreen				
	objects as defined in the "Actuated			preemptMinimumWalk				
	Signal Controller Object			preemptEnterPedClear				
	Definitions," NTCIP 1202:1996							
	Implement all mandatory objects of			preemptTrackGreen				
	all optional conformance groups as			preemptDwellGreen				
	defined in NTCIP 1202:1996:			preemptMaximumPresence				
	Preempt Conformance Group			preemptTrackPhase				
				preemptDwellPhase				
	Not Required ⁷	N/A		preemptDwellPed				

⁷ Optional object not required by DMS 11170-TSC

	DMS 11170-TSC							
	Requirements Traceability Matrix							
User Need	Requirement	TxDOT DMS 11170 Reference	NTCIP Object Support		Test Procedure Identifier			
	Implement all mandatory objects of all optional conformance groups as defined in NTCIP 1202:1996: Preempt Conformance Group	NTCIP Compliance		preemptExitPhase				
	Implement the following optional objects as defined in the "Actuated Signal Controller Object Definitions," NTCIP 1202:1996			preemptState				
	Not Required ⁸	N/A		preemptTrackOverlap				
				preemptDwellOverlap				
				preemptCyclingPhase				
				preemptCyclingPed				
				preemptCyclingOverlap				
				preemptEnterYellowChange				
				preemptEnterRedClear				
				preemptTrackYellow Change				
				preemptTrackRedClear				
	Implement the following optional	NTCIP	preemptControlTable		PRL-TC001			
	objects as defined in the "Actuated	Compliance		preemptControlNumber				
	Signal Controller Object Definitions," NTCIP 1202:1996			preemptControlState				
Ring								

⁸ These objects were added in NTCIP 1202 v02.18.

DMS 11170-TSC Requirements Traceability Matrix					
User Need	Requirement	TxDOT DMS 11170 Reference	NTCIP Object Support		Test Procedure Identifier
	Implement all mandatory objects of all optional conformance groups as	NTCIP Compliance	maxRings maxSequences		PRL-TC001 PRL-TC001
	defined in NTCIP 1202:1996: Ring Conformance Group		sequenceTable sequenceNumber sequenceRingNumber		PRL-TC001
			maxRingControl Groups		PRL-TC001
			ringControlGroup Table	ringControlGroupNumber ringControlGroupStopTime	PRL-TC001
	Implement the following optional objects as defined in the "Actuated	-		ringControlGroupForceOff	
	Signal Controller Object Definitions" NTCIP 1202:1996			ringControlGroup MaxInhibit	
	Implement all mandatory objects of all optional conformance groups as defined in NTCIP 1202:1996: Ring Conformance Group			ringControlGroupPed Recycle	
	Not Required ⁹			ringControlGroupRedRest	
				ringControlGroupOmitRed Clear	
	Not Required ¹⁰	N/A	ringStatusTable		PRL-TC001

⁹ These optional objects are not required by DMS 11170-TSC. ¹⁰ This table and object were added in NTCIP 1202 v02.18.

DMS 11170-TSC						
Requirements Traceability Matrix						
User Need	Requirement	TxDOT DMS 11170 Reference	NTCIP Object Support		Test Procedure Identifier	
				ringStatus	Seg-TC001 DetOps-TC001 – DetOps-TC018, IM-TC0001	
Chann	el	1	1		1	
	Implement all mandatory objects of	NTCIP	maxChannels			
	all optional conformance groups as	Compliance	channelTable		PRL-TC001	
	defined in NTCIP 1202:1996:			channelNumber		
	Channel Conformance Group			channelControlSource		
				channelControlType		
				channelFlash		
				channelDim		
			maxChannelStatusGroups		PRL-TC001	
			channelStatusGroupTable		PRL-TC001	
				channelStatusGroupNumber		
				channelStatusGroupReds		
				channelStatusGroupYellows		
				channelStatusGroupGreens		
Overlap						
	Implement all mandatory objects of	NTCIP	maxOverlaps		PRL-TC001	
	all optional conformance groups as	Compliance	overlapTable		PRL-TC001	
	defined in NTCIP 1202:1996:			overlapNumber		
	Overlap Conformance Group			overlapType		
				overlapIncludedPhases		
				overlapModifierPhases		

DMS 11170-TSC							
Requirements Traceability Matrix							
User	Requirement	TxDOT	NTCIP Object Support		Test Procedure		
Need		DMS 11170			Identifier		
		Reference					
				overlapTrailGreen			
				overlapTrailYellow			
				overlapTrailRed			
			maxOverlapStatus		PRL-TC001		
			Groups				
			overlapStatusGroup		PRL-TC001		
			Table	overlapStatusGroupNumber			
				overlapStatusGroupReds			
				overlapStatusGroupYellows			
				overlapStatusGroupGreens	DetOps-TC001 -		
					DetOps-TC018		
TS 2 P	ort 1				· •		
	Implement all mandatory objects of	NTCIP	maxPort1Addresses		PRL-TC001		
	all optional conformance groups as	Compliance	port1Table		PRL-TC001		
	defined in NTCIP 1202:1996: TS 2		-	port1Number			
	Port 1 Conformance Group			port1DevicePresent			
	-			port1Frame40Enable			
				port1Status			
				port1FaultFrame			
Block (Block Objects						
	Not Required ¹¹	N/A	ascBlockGetControl				
	_		ascBlockData				
			ascBlockErrorStatus				

¹¹ These objects were added in NTCIP 1202 v02.18.

DMS 11170-TSC					
Requirements Traceability Matrix					
User	Requirement	TxDOT	NTCIP Object Support	Test Procedure	
Need		DMS 11170		Identifier	
		Reference			

Traceable to NTCIP 1201-GLO						
Time Management						
Implement all mandatory objects of	NTCIP	globalTme		TAD-TC0001		
all optional conformance groups as	Compliance	globalDayLight		TAD-TC0001		
defined in "Global Object		Savings				
Definitions," NTCIP 1201:1996:						
Time Management Conformance						
Group						
Deprecated objects ¹²		globalLocalTimeDifferential		TAD-TC0001		
Time Base Event Schedule	Time Base Event Schedule					
Implement all mandatory objects of		maxTimeBase ScheduleEntries timeBaseSchedule		PRL-TC001		
all optional conformance groups as						
defined in "Global Object				PRL-TC001		
Definitions," NTCIP 1201:1996:		Table	timeBaseSchedule			
Timebase Event Schedule		-	Number			
Conformance Group			timeBaseScheduleMonth			
			timeBaseScheduleDay			
			timeBaseScheduleDate			
			timeBaseScheduleDayPlan			
Not Required ¹³	N/A	timeBaseSchedule				
		Table-status				

¹² This object was deprecated in NTCIP 1201 v02.26.
¹³ This object was added in NTCIP 1201 v02.26.

DMS 11170-TSC						
Requirements Traceability Matrix						
User Need	Requirement	TxDOT DMS 11170 Reference	NTCIP Object Support		Test Procedure Identifier	
	Implement all mandatory objects of	NTCIP	maxDayPlans		PRL-TC001	
	all optional conformance groups as	Compliance	maxDayPlanEvents		PRL-TC001	
	defined in "Global Object		timeBaseDayPlan		PRL-TC001	
	Definitions," NTCIP 1201:1996:		Table	dayPlanEventNumber		
	Timebase Event Schedule			dayPlanHour		
	Conformance Group			dayPlanMinute		
				dayPlanActionNumber OID		
			dayPlanStatus			
	Not Required ¹⁴	N/A	controller-		TAD-TC0001	
			standardTimeZone			
	Not Required ¹⁵		controller-localTime			
Databa	Database Management and Configuration Man					
	Implement all mandatory objects of all optional conformance groups as defined in "Global Object Definitions," NTCIP 1201:1996:	NTCIP Compliance	dbCreateTransaction		DCT-TC0001	
	Database Management Conformance Group					
	Deprecated objects ¹⁶		dbErrorType			
			dbErrorID			
			dbTransactionID			

¹⁴ This object was added in NTCIP 1201 v02.26.
¹⁵ This object was added in NTCIP 1201 v02.26.
¹⁶ These objects were deprecated in NTCIP 1201:1996 v01.10.
	DMS 11170-TSC Requirements Traceability Matrix					
User Need	Requirement	TxDOT DMS 11170 Reference	NTCIP OF	oject Support	Test Procedure Identifier	
	Implement the following optional objects as defined in the "Global Object Definitions," NTCIP 1201:1996 ¹⁷		dbMakeID			
	Implement all mandatory objects of all optional conformance groups as defined in "Global Object Definitions," NTCIP 1201:1996: Database Management Conformance Groupdb		dbVerifyStatus dbVerifyError		DCT-TC0001	
Config	uration (Global)				·	
	Implement all mandatory objects of all mandatory conformance groups as defined in the "Global Object		globalSetIDParmeter		(ASC)GloCon- TC002	
	Definitions," NTCIP 1201:1996: Global Configuration Conformance Group		globalMaxModules		(CCTV)Config- TC001, (ASC) GloCon- TC001 PRL-TC001	
			globalModuleTable		PRL-TC001	
				moduleNumber moduleDeviceNode moduleMake moduleModel	(ASC) GloCon- TC001	

 Table 4. TSC Requirements Traceability Matrix with Test Procedures (continued).

¹⁷ This object was deprecated in NTCIP 1201:1996 v01.10.

	DMS 11170-TSC Requirements Traceability Matrix					
User Need	Requirement	TxDOT DMS 11170 Reference	NTCIP Object Support		Test Procedure Identifier	
				moduleVersion moduleType		
	Not Required ¹⁰	N/A		controller-baseStandards		
Report		-1	1		1	
	Implement all mandatory objects of	NTCIP	maxEventLogConfigs		PRL-TC001	
	all optional conformance groups as	Compliance	eventLogConfigTable		PRL-TC001	
	defined in the "Global Object			eventConfigID	RLD-TC001 -	
	Definitions," NTCIP 1201:1996:			eventConfigClass	RLD-TC0008	
	Report Conformance Group			eventConfigMode	-	
				eventConfigCompareValue	-	
				eventConfigCompareValue2		
	Implement the fellowing optional	_		eventConfigLogOID	-	
	Implement the following optional			eventConfigAction	-	
	objects as defined in the "Global			eventeoningAction		
	Object Definitions," NTCIP					
	1201:1996		_		-	
	Not Required ¹⁹	N/A		eventConfigStatus		
	Implement all mandatory objects of	NTCIP	maxEventLogSize		PRL-TC001	
	all optional conformance groups as	Compliance	eventLogTable		PRL-TC001	
	defined in the "Global Object	-		eventLogClass	RLD-TC001 -	
	Definitions," NTCIP 1201:1996:			eventLogNumber	RLD-TC0008	
	Report Conformance Group			eventLogID		
	r			eventLogTime	-	
				eventLogValue		

 Table 4. TSC Requirements Traceability Matrix with Test Procedures (continued).

¹⁸ This object was added in NTCIP 1201 v02.26.
¹⁹ This object was added in NTCIP 1201 v02.26.

[DMS 11170 TSC							
	Requirements I raceability Matrix							
User	Requirement	TxDOT	NTCIP O	bject Support	Test Procedure			
Need		DMS 11170			Identifier			
		Reference						
			maxEventClasses		PRL-TC001			
			eventClassTable		PRL-TC001			
				eventClassNumber	RLD-TC001 -			
				eventClassLimit	RLD-TC0008			
		-		eventClassClearTime				
	Implement the following optional			eventClassDescription				
	objects as defined in the "Global							
	Object Definitions," NTCIP							
	1201:1996							
	Implement all mandatory objects of			eventClassNumRowsInLog				
	all optional conformance groups as							
	defined in the "Global Object							
	Definitions," NTCIP 1201:1996:							
	Report Conformance Group							
	Not Required ²⁰	N/A	numEvents					
STMP		-						
	Implement all mandatory objects of	NTCIP	dynamicObject					
	all optional conformance groups as	Compliance	Persistence					
	defined in the "Global Object	-						
	Definitions," NTCIP 1201:1996:							
	STMF (STMP) Conformance Group							

 Table 4. TSC Requirements Traceability Matrix with Test Procedures (continued).

²⁰ This object was added in NTCIP 1201 v02.26.

DMS 11170-TSC					
Requirements Traceability Matrix					
User	Requirement	TxDOT	NTCIP Object Support	Test Procedure	
Need		DMS 11170		Identifier	
		Reference			

PMPP					
	Implement all mandatory objects of	NTCIP	maxGroupAddresses		PRL-TC001
	all optional conformance groups as	Compliance	hdlaGroupAddross		222 TC012 and
	Definitions," NTCIP 1201:1996:		Table	hdlcGroupAddressIndex	232-TC015 and 232-TC014
	PMPP Conformance Group			hdlcGroupAddress	
Securit	У				
	Implement all mandatory objects of	None	communityName		(ASC) Security-
	all mandatory conformance groups as		Admin		TC001
	defined in the "Global Object		communityNamesMax		(ASC) Security-
	Definitions," NTCIP 1201:1996:				TC002
	Security Conformance Group		communityNameTable		(ASC) Security-
				communityNameIndex	TC002
				communityNameUser	
				communityNameAccess	
				Mask	

Test Plan and Documentation

Since a task of developing NTCIP test procedures for one of the ITS field devices can be a significant project in and of itself, the researcher found that following some of the recommendations in the Institute of Electrical and Electronics Engineers (IEEE) Std. 829 – IEEE Standard for Software Test Documentation are useful (3). Prior to actually writing procedures, the IEEE standard suggests the development of an overall plan, one or more test design specifications, and test case specifications. The overall plan conveys the scope, approach, resources, and schedule of testing activities. Its primary purpose is to present a high-level view of the project to inform all interested parties. The test design specifications provide a more detailed view of the testing project. A test engineer's supervisor and any group such as a project monitoring committee uses the test design specifications to make sure that a test engineer understands the projects and is addressing what is needed. This serves as part of the validation step in the project development. Test case specifications then outline individual test cases that verify specific features and functions of an implementation undergoing testing. The test case specifications provide additional oversight but primarily help a test engineer organize and plan the specifics of each test case before committing to code or formal definition. These types of documents address the planning aspects of a testing project.

Full development of these documents is beyond the scope of this TxDOT research project. In the case of the CCTV test procedures, the researcher capitalized on test procedures already in the public domain, and upfront planning did not appear to be necessary. For the traffic signal controllers, the researcher had a test case specification from a previous project that was appropriate. A previous test design specification was somewhat appropriate, and the researcher believes that an update that references the TxDOT department material specification DMS 11170-TSC, Fully Actuated, Solid-State Traffic Signal Controller Assembly provides useful information (2). Appendix F contains a Test Design Specification and a Test Case Specification for addressing all the functions of a traffic signal controller. It uses IEEE Std. 829 – IEEE Standard for Software Test Documentation as a guide on the organization and contents.

A number of state departments of transportation are adopting International Organization for Standardization (ISO) 9000 in order to improve quality (21). The following test design

59

specification and test case specification are two types of document examples that would satisfy some of ISO 9000 requirements.

Traffic Signal Controller Test Procedures

Using the information in Appendix F as the basis for development, the researcher provides the details of a limited set of test cases and their test procedures in Appendix G. The following traffic signal controller test cases focus on:

- PRL Information
- Four-Phase Diamond Sequencing
- Four-Phase Diamond Detector Operations
- Global Configuration
- Security

Some additional test cases also look primarily at the functionality. These test cases are based on user needs in that they look at what services a user would expect or perform on a conformant device. These test cases consist of:

- Managing Phase Configuration and Initialization
- Retrieving Phase Status Information
- Retrieving Manufacturer Information
- Setting Up and Retrieving Log Data
- Retrieving System Status Information
- Setting Up and Executing Timebase Events
- Setting Time and Date

Traffic Signal Controller Test Results

The suggestion for reporting traffic signal controller test results is to use the same method as illustrated for CCTV. The PRL from the NTCIP 1202-ASC standard serves as a template and the ASC PRL Information prequalification test script (see Appendix G) outputs the results onto that form (*16*). The completed test results would be similar to what appears in Appendix D. In the case of other test procedures, the NTCIP test procedure itself serves as the template and the related test script transcribes the results onto the template.

The researcher's definition of the traffic signal controller test procedures do not follow the method as prescribed in NTCIP 8007 – Testing and Conformity Assessment Documentation within NTCIP Standards Publications (NTCIP 8007-TEST) (22). After reviewing the result reports for the CCTV test procedures, the researcher believes that showing results for every test step does not enhance readability and may lead to overlooking something important. For example, Figure 2 shows "Pass" for every test step. It may be clearer to show only "Pass" for the critical steps as shown in Figure 3. Any test step that does fail would show a "Fail," and the rest of the indications in the test procedure template would be unchanged.

Test Step Number	Test Procedure	Results
1	SET labelText. <alarmclabindex> = <alarmclabtext1> labelHeight.<alarmclabindex> = <alarmclabheight1> labelColor.<alarmclabindex> = <alarmclabcolor1> labelStartRow.<alarmclabindex> = <alarmclabstartrow1> labelStartColumn.<alarmclabindex> = <alarmclabstartcolumn1></alarmclabstartcolumn1></alarmclabindex></alarmclabstartrow1></alarmclabindex></alarmclabcolor1></alarmclabindex></alarmclabheight1></alarmclabindex></alarmclabtext1></alarmclabindex>	
2	SET alarmLabelIndex.0 to <alarmclabindex> 00 00 00 00 00 00 00</alarmclabindex>	
3	USER VERIFY that no labels are being shown.	Pass
4	SET alarmLatchClear.0 to 0x00	
5	Turn on the alarm and USER VERIFY the label for the alarm is shown.	
6	GET alarmStatus.0 and alarmLatchStatus.0	
7	VERIFY RESPONSE VALUE alarmStatus = 0x80 alarmLatchStatus = 0x80	Pass
8	SET alarmLatchClear.0 to 0x00	Pass

Figure 3. Test Results Indicating Critical Results.

DETECTOR REQUIREMENTS

During the course of developing test procedures for the four-phase detector operations as defined in DMS 11170-TSC, the researcher ran across a situation where the requirements may not be addressing an operational need (2). The description of how detectors 1 and 5 are to operate does not address the situation in which there is a call for service on one of the left turns when a controller is resting in the opposing through green. Signage and the physical geometry of an actual intersection may explain it, but when an implementation follows the requirements, a call does not register.

CHAPTER 3: DEVELOPING ADDITIONAL TEST PROCEDURES

INTRODUCTION

In order to provide TxDOT with a sense of the effort one needs to develop test procedures, the researcher presents Table 5. The table references the standards that would require test procedures, statistical information about the number of object definitions and tables to gauge complexity, information about NTCIP efforts to develop test procedures, and the researcher's estimates to implement test procedures in a suitable TxDOT testing environment. The number of object definitions and tables comes from Version 1 of each standard. Typically, Version 2 of each standard contains more objects.

The estimates come from experience in deriving the test procedures in this report, previous test procedure development in other projects, and conversations with several NTCIP working group chairs, technical editors, and/or consultants. The test procedures under consideration are not exhaustive. For example, virtually any object can trigger the logging of an event. Rather than test all possible triggers, the procedures would look at typical objects of each data type to gauge overall correctness of the function.

In general, read-only objects represent either preset values or status information with status information correlating to the overall complexity. Preset values typically define the number of instances associated with entries in a table. A test procedure to check presets is trivial. Status information, on the other hand is relatively complex to test. This testing usually involves the setup of conditions to invoke individual states of a status object. The number of status objects is approximately equal to the number of read-only object definitions minus the number of tables.

Standard	Object Definitions		Tables	NTCIP Test	Effort for Test	
	Read-Only	Read-Write	Total		Procedures	Procedures
1103-TMP	77	35	112	7		4-6 months
1201-GLO	46	27	73	8	Planned	2-3 months
1202-ASC	85	98	183	21		8-10 months
1203-DMS	79	53	132	7		2-4 months
1204-ESS	83	7	90	4	Planned	2 months
1205-CCTV	35	54	89	2		2.5 months
1206-DCM	151	79	230	30		8-10 months
1207-RMC	59	143	202	19		5-7 months
1208-SW	22	37	59	8		3 months
1209-TSS	27	18	45	8		4-6 months
1210-FMS	77	104	181	23		7-9 months
1211-SCP	43	5	48	4	Planned	2 months
1213-ELMS	26	63	89	9		3 months

Table 5. NTCIP Standard Statistics and Test Procedure Efforts.

ITS FIELD DEVICE ESTIMATES

The following sections provide a description of the individual estimates. The estimates do not include any effort to locate and acquire an implementation in order to validate the test procedures. One should also understand that additional hardware is often necessary to check the functionality of an implementation. For example, a data collection and monitoring device would need various types of sensors to produce valid or runtime-status information values. In the case of signal control and prioritization, a system consists of a request generator, request server, and traffic signal controller. All three components would need to be on hand to fully test a system.

NTCIP 1103-TMP

The NTCIP 1103 Transportation Management Protocols (NTCIP 1103-TMP) standard contains statistical objects related to STMP and SFMP, STMP configuration objects, the event

and report objects, and the community name objects (23). The effort to define test procedures for the STMP statistical objects is moderate because it requires the creation of error conditions. Although some related procedures exist, SFMP procedures will require design time. Complicating the development of SFMP procedures is the fact that testing tools do not support the protocol, and implementations are not apparent. Therefore, SFMP test procedures are not part of the estimate. The estimate for the level of effort for STMP configuration objects is relatively low because there has been preliminary development. This estimation assumes that the procedures will address the encodings for only a sample of the various object types. A more robust procedure that looks at several dozen possible typical messages would take considerably longer. The effort for event and report objects test procedures is also relatively low because there has been preliminary development. The trap management object that will appear in NTCIP 1103-TMP Version 2 would double the effort and push the estimate out to the 6-month mark (23).

NTCIP 1201-GLO

The NTCIP 1201 – Global Object Definitions (NTCIP 1201-GLO) standard contains objects that relate to general functions within ITS field devices (7). It covers non-device specific functions like time and date, auxiliary input and output, and general scheduling information. The effort for global objects procedures is relatively low because there has been some preliminary development. There is a strong likelihood that an NTCIP working group will add these to the standard itself. The researcher estimates the effort for developing NTCIP 1201-GLO test procedures at 2 to 3 months.

NTCIP 1202-ASC

The effort to develop test procedures for NTCIP 1202 – Object Definitions for Actuated Signal Controller Units (NTCIP 1202-ASC) ranges from 8 to 10 months (*16*). There are private industry efforts to develop NTCIP 1202-ASC test procedures. However, there is some reluctance to place these procedures in the public domain. A proposal to add them to the NTCIP standard was not approved. Assuming these private industry procedures are not available, then the effort will be somewhat lengthy. This estimate assumes a limited number of test cases wherein the number of combinations and permutations is minimal. The estimate includes 1 month to address three-phase diamond and dual four-phase operations. With these assumptions, the estimate for NTCIP 1202-ASC test procedures is between 8 and 10 months.

NTCIP 1203-DMS

The estimate for NTCIP 1203-DMS test procedures ranges from 2 to 4 months. The unknown variable is support for Version 2 (5). Test procedures for Version 1 are already in the public domain. Assuming these test procedures are acceptable and they simply need updating and reformatting, then the effort is about 2 months. From a conversation with the editor of the NTCIP-1203 standard, Version 2 test procedures may be currently under development as part of a Virginia DOT project (5). If these are to serve as the basis for TxDOT procedures, the effort estimate is 4 months.

NTCIP 1204-ESS

For NTCIP 1204 – Environmental Sensor Station Interface Standard (NTCIP 1204-ESS), the estimate is 2 months (24). In a conversation with the NTCIP program manager on August 10, 2006, the NTCIP 1204-ESS working group has a notice-to-proceed with adding test procedures to the standard (24). Assuming these are acceptable to TxDOT, then the effort will be translating them into appropriate scripts. The researcher's estimate for NTCIP 1204-ESS test procedures is 2 months.

NTCIP 1205-CCTV

This research report includes a set of test procedures for NTCIP 1205 – Object Definitions for Closed Circuit Television (CCTV) Camera Control. Omitting the effort to locate and acquire NTCIP compliant hardware to validate the procedures, the effort to reformat the Enterprise test procedures into the NTCIP 8007 – Testing and Conformity Assessment Documentation within NTCIP Standards Publications (NTCIP 8007-TEST) format and convert them to SimpleTester^{TM 21} for NTCIP test scripts was approximately 2.5 months (22).

NTCIP 1206-DCM

NTCIP DCM working group chair's estimate for test procedures related to NTCIP 1206 – Object Definitions for Data Collection and Monitoring (DCM) Devices (NTCIP 1206-DCM) is between 5 and 7 months (25). Given the number of status information objects, however, this appears to be conservative. The researcher believes that describing and defining the test

²¹ SimpleTesterTM is a trademark of SimpleSoft, Incorporated, Mountain View, California.

conditions will take considerable effort. Even without a working knowledge of a DCM device, the number of status information objects is almost double that of other devices with a high level of complexity. For this reason, the researcher's estimate is 3 months longer than the chair's estimate and is put at 8 to 10 months.

NTCIP 1207-RMC

The estimate for NTCIP 1207 – Object Definitions for Ramp Meter Control (RMC) Units (NTCIP 1207-RMC) comes from the NTCIP RMC working group chair (*26*). When one compares the number of status information objects to other standards, this estimate is somewhat high. However, in the case of RMC devices, the researcher defers to the judgment of the working group chair whose estimate is 5 to 7 months.

NTCIP 1208-SW

Given the least number of status information objects of any standard, the estimate for NTCIP 1208 – Object Definitions for Closed Circuit Television (CCTV) Switching (NTCIP 1208-SW) is put at 3 months (27).

NTCIP 1209-TSS

The NTCIP 1209 – Data Element Definitions for Transportation Sensor Systems (NTCIP 1209-TSS) estimate is between 4 and 6 months (28). The number of objects in Table 5 comes from Version 1 of NTCIP 1209-TSS. This number of objects is at the lower end of the scale when considering other NTCIP standards. However, the working group chair cautions that Version 2 is going to include additional object support for machine vision and vehicle classification. Since these two areas are of interest to TxDOT, the upper end of the 4 to 6 month estimate is for support of Version 2.

NTCIP 1210-FMS

The basis for the estimate of NTCIP 1210 – Field Management Stations - Part 1: Object Definitions for Signal System Masters (NTCIP 1210-FMS) test procedures is the current working group draft (29). Even though the final object count will likely change before Version 1 has full approval, the researcher and the FMS working group consultant are in general agreement

67

on the amount of effort to develop a set of test procedures. The estimate for FMS test procedures is 7 to 9 months.

NTCIP 1211-SCP

Even though the NTCIP 1211 – Object Definitions for Signal Control and Prioritization (NTCIP 1211-SCP) standard deals with the potential for a system consisting of multiple physical entities, the effort to implement a set of procedures for TxDOT should be relatively small (30). This is because an NTCIP project plan to add them to the standard has approval. If these NTCIP test procedures meet with TxDOT's approval then the effort will be translating them into appropriate scripts. This estimate is 2 months.

NTCIP 1213-ELMS

The estimate for NTCIP 1213 – Objects Definitions for Electrical and Lighting Management Systems (NTCIP 1213-ELMS) is 3 months (*31*). This estimate comes from the number of object definitions in the standard and a general discussion with the working group chair. Given the small number of status information objects, defining the procedures should be relatively easy. As stated in the introduction, however, the bigger issue will be to identify and acquire field devices to verify the procedures.

CHAPTER 4: APPLYING TESTING PROCEDURES

TESTING PROCESSES

TxDOT documents and specifications describe two testing processes. Tex-1170-T – Sampling and Environmental Testing of Traffic Signal Controller Assemblies: Traffic Signal Controllers and Conflict Monitors (Tex-1170-T) defines an internal TxDOT process that applies to traffic signal controllers and cabinets (*32*). The second process comes from Special Specification 6504 – Testing, Training, Documentation and Warranty (SS 6504) (*33*). This specification describes a testing process that is the responsibility of a contractor to perform. The following headings describe the two processes and apply different testing techniques to each of them. A draft information report, NTCIP 9012 – Testing and Conformity Assessment User Guide for NTCIP Field Devices and Center-to-Field Communications (NTCIP 9012-TG) provides general background and guidance to an agency defining a testing process/program for testing devices that incorporate NTCIP Standards (*34*).

INTERNAL TXDOT TESTING PROCESS

Figure 4 illustrates the process in Tex-1170-T. For a bidder to provide equipment under a contract, the equipment must usually be on TxDOT's Qualified Products List. To qualify for the list, equipment must pass QPL testing. Assuming that it is on the list, a bidder provides equipment that meets any contract-specific specifications or provisions to the district office. The district sends a sample of the equipment to the Traffic Operations Division for environmental and, possibly, QPL testing. This round of QPL testing only takes place if the sample is not already on the QPL list. At the same time, the district checks and configures the equipment for its specific installation requirements. Once the equipment passes these checks, it undergoes a field check to see if it operates properly in the system.



Figure 4. TxDOT Testing Activities.

Prequalification Testing

Prior to performing QPL testing, a tester should review a manufacturer's completed Protocol/Profile Implementation Conformance Specification (PICS) or perform the Protocol/Profile Requirements List Information Test Procedure in Appendix C to produce a PICS. NTCIP standards that deal with object definitions for a device usually provide a PRL that has one or more tables summarizing object definitions in the Management Information Base. After filling out a PRL with the information about support for objects or features, indicating the range of supported values, and showing values for indexed items, it becomes a PICS. Using the PRL/PICS as the basis for a report produces a uniform and consistent manner in which to compare similar devices.

The script of the PRL test procedure in Appendix C creates a PICS for a CCTV control unit. The test procedure creates the PICS by:

- retrieving minimum project requirements values and indexing parameters,
- checking that all objects in a MIB can be read,
- performing a set operation on writeable objects with a sampling of values to check an object's range, and
- recording the information on a PRL that appears in the NTCIP standards.

A SimpleTester TM for NTCIP script automates the process. The script for CCTV takes only several minutes to run. Appendix D shows a sample completed PICS.

A review of a PICS serves the purpose of prequalification testing or one of the first steps in QPL testing. One can use the PICS to determine if it is worth spending the time to test an implementation extensively. If the PICS indicates that an implementation does not support a required function or the required number of instances of an object, testing the functions and objects that are implemented would not serve any purpose.

QPL Testing

The focus of QPL testing is compliance to TxDOT specifications for a specific device. Compliance to the specification entails a 100 percent check of all requirements. It is exhaustive and covers:

- hardware design,
- conformance to external standards,
- functionality, and
- documentation.

QPL testing uses a device's specification as a guide to ensure that equipment meets all of the requirements. While testing for compliance to NTCIP does not address hardware design and documentation, it can address conformance to some external standards and functionality. Compliance to TxDOT specifications does not necessarily have to be a separate process. Consider the Absolute Pan Motion test procedure in Appendix C that tests one of the panning motion functions defined in NTCIP 1205-CCTV. The procedure consists of:

- 1. Move the camera to a predefined position.
- 2. Ask the user to verify that the camera moved to that position.
- 3. Internally verify that the camera moved to that position.
- 4. Move the camera to another predefined position.
- 5. Ask the user to verify that the camera moved to that position.
- 6. Internally verify that the camera moved to that position.

The intent of the procedure is to check the conformance of three NTCIP objects. As it is stated, the procedure could be suitable for all the phases of the TxDOT testing process. By referencing externally defined positions, the procedure is suitable for statewide, district, and one-

time use because the positions values are customizable to the individual requirements. The means by which the user verifies the new positions could also customize it to TxDOT QPL or to configuration and system testing. In the case of QPL, one could quantitatively measure the angle. For example, during QPL, one could use a protractor to define reference points to measure angular positions. For configuration and system testing, simply estimating the angle may be sufficient.

Configuration Testing

The focus of configuration testing is on meeting project-specific requirements. Using an approach where variables come from an external source allows a tester to customize the test procedure to his or her needs. Using the CCTV Absolute Pan Motion test procedure, the test could use pan position values that come from the project specification or the geometry of the intended location.

During configuration testing, one common practice is to run a test repetitively to simulate actual field operations and stress the unit. In this case, a simple modification to the procedure might be to insert additional looping instructions. Using the CCTV Absolute Pan Motion test procedure as an example, the procedure would look something like the following:

- 1. For N = 1 to predefined reiterations
 - 1. Move the camera to a predefined position
 - 2. If predefined reiterations = 1 then
 - i. Ask the user to verify that the camera moved to that position
 - 3. Internally verify that the camera moved to that position
 - 4. Move the camera to another predefined position
 - 5. If predefined reiterations = 1 then
 - i. Ask the user to verify that the camera moved to that position
 - 6. Internally verify that the camera moved to that position
- 2. Next reiteration

Defining the number of predefined reiterations as 1 makes it suitable for QPL testing where a tester physically verifies the positions. Defining the number of predefined reiterations as 100 could make it more appropriate for configuration testing where the emphasis is on stressing the equipment.

Sample Environmental System Testing

During environmental testing, a test chamber subjects a device to temperature and linevoltage extremes that may occur in actual field operation. The device should continue to run and time intervals correctly under combinations of high and low environmental conditions. Although NTCIP test procedures are not directly applicable to environmental tests, the browser functions of one of the active testing tools could find usage in changing the configuration of a device without having to enter the test chamber. A test procedure that checks a traffic signal controller's NTCIP 1202-ASC download features could configure a controller with a standard set of configuration controls and interval times (*16*). There is nothing to prevent any NTCIP functional test procedure from also being run while a device is in an environmental chamber.

System Testing

During system testing, a management application connects to a device either while the device is still at a facility or after installation in the field location. NTCIP test procedures have two uses during system testing. They can serve as a test of the management application to show that it supports the functionality expressed by the NTCIP objects. An instrumentation testing tool can also validate that a device responds correctly even though communications take place over an agency's communications infrastructure.

CONTRACTOR TESTING PROCESS

Figure 5 illustrates the process in SS 6504. During design approval testing, the contractor either runs environmental tests directly or has an independent testing laboratory conduct them. During demonstration testing (conducted prior to installation), the contractor performs a physical inspection of the equipment and performs operational tests to ensure compliance to the specifications (*33*). After the contractor installs the equipment but before connection to any other components of the system, stand-alone testing verifies functional operations. After connection, system integration testing demonstrates that all control and monitor functions are operating properly. TxDOT personnel do not perform the tests, but a reserve clause in the specification allows someone from TxDOT to observe the tests. The TxDOT engineer is responsible for overall approval and final acceptance.

73



Figure 5. Contractor Testing Activities.

NTCIP test procedures find application in contractor testing in much the same way as they do in the internal TxDOT testing process. NTCIP test procedures can add standardized functional test procedures during the design approval testing. During the operation-testing portion of the demonstration test, the NTCIP test procedures cover the functionality expressed by the NTCIP object definitions. The NTCIP object definitions, in turn, cover most, if not all, the functionality of a device. During the stand-alone test, the NTCIP test procedures provide not only standardized tests but also a means to conduct the testing. Although NTCIP standards do not formalize a set of test procedures as yet, the de facto procedures are undergoing peer review. With testing tool support coming from many sources, contractors do not have to develop their own software to conduct the test. There is also the benefit that by prescribing the use of NTCIP procedures, the engineer gains a greater understanding of what is taking place. During system integration testing, most of the communications driving the functionality of a device come from a management application. The NTCIP test procedures would not apply in this case. However, the instrumentation testing tools that support NTCIP could provide an independent means to verify that a management application is issuing the proper commands and that the device is responding correctly. SS 6504 also requires that contractors submit test procedures and data forms to the TxDOT engineer prior to any actual testing for his or her approval. NTCIP test

procedures provide more consistency in the documentation and are easier to understand (*33*). Adopting the reporting format as suggested in this research should make it easier for engineers to interpret the report and make the reports consistent across all contractors.

A TxDOT engineer or other TxDOT personnel could verify contractor testing by the use of an instrumentation-testing tool. This type of testing tool can provide independent verification that a contractor is performing the test procedures as described and that the results are as they should be. The same information would also be available when used during system integration or actual system operation. An instrument-testing tool provides a message view of exchanges between two parties. The tool can show what a management or test application sends. It will also show the responses from the device under test. Rather than showing a string of bytes that appear on the wire, a tool can decode the bytes into human readable parameters that indicate the type of command sent, the names of NTCIP objects involved, and the values of the objects. Figure 6 is an example of how one instrument-testing tool presents information to a user.



Figure 6. Instrumentation Testing Tool Information Example.

The display shows the response of a traffic signal controller to a get request asking for a description of the device. The response shows that the STMPv1 protocol is used, the community name or access code was "administrator," the packet type was a "get response," and there were no errors. The name of the NTCIP object that contains the description is "sysDescr" and that the value is "ETCS EPAC300, 2070N, OS …" Those familiar with traffic signal controllers will recognize this as an Eagle Traffic Control System – EPAC 300 series signal controller running on ATC 2070N hardware.

CONFIGURATION MANAGEMENT AND VERSION CONTROL

As manufacturers implement the NTCIP standards and users deploy equipment in the field, the standards development groups receive feedback about problems and requests for enhancements. This invariably leads to a revision of the standards. Unless one takes specific

steps to avoid the situation, it is possible to have systems consisting of equipment conformant to different versions of a standard. For example, the initial attempts to define object definitions related to time-of-day resulted in several alternate but valid ways of how to treat the objects. The dates for implementing daylight saving time are now different. Revisions to the standard reflect corrections and the new rules. Some manufacturers are implementing the newest versions of the standard. Although NTCIP strives to make revisions backwards compatible, it is not always possible. Unless a management application is up-to-date, it will not be able to deal with any changes.

Any specification that references an NTCIP standard should specify the specific version and revision date. Since some NTCIP standards deal with functionality applicable to all field equipment (NTCIP 1201-GLO, for example), specifications of various equipment may need to be consistent in this regard as well (7). If one specification calls for a specific version number, interoperability problems may crop up if other specifications reference another version. An agency should also maintain records containing location and version information. These two steps will minimize incompatibility problems and help in understanding the impacts of any future upgrade.

CHAPTER 5: TRAINING

INTRODUCTION

This chapter presents two training course outlines. The first looks at testing from an NTCIP perspective and the second outline looks at testing from a TxDOT perspective. The first outline begins with an explanation of the difference between conformance testing in relation to the NTCIP standards and compliance testing in relation to TxDOT specifications. The outline then addresses background information on the NTCIP standards in order to put testing into perspective. There is a discussion on the two types of standards: data dictionaries and protocols/profiles. The background also covers two NTCIP standards related to testing, and there is also a reference to the NTCIP framework on how to combine standards to build an implementation. The next part of the first outline deals with terminology and techniques. The last part covers how to interpret results. Different groups and organizations have different methods for reporting test results. Testing tools also incorporate some type of report. If a large agency, such as TxDOT, needs to examine these reports, it takes a bit of understanding to draw meaningful conclusions.

The second outline looks at testing from the TxDOT perspective. Given enough time and resources, one could fully test everything covered by NTCIP and the functionality associated with the object definitions. However, there is usually a lack of resources, and time is always at a premium. These limitations introduce the subject of risk management. Risk management looks at techniques to minimize the amount of testing and still maintain a high level of confidence that an implementation is correct. The outline then looks at what to test and the general techniques to use. The next part examines various tools that are available. It explains the three types of tools: active, emulator, and instrumentation and covers some of their characteristics. The last part of the outline discusses configuration management. Without specifying specific standards versions, one can expect interoperability issues. The chapter ends with a suggested evaluation form.

AUDIENCE

The level of detail in this training course is meant for someone responsible for planning or carrying out testing activities. Most of the material is non-technical in nature. However, a

77

person may need to ensure that messages use the proper formatting, encoding, and NTCIP protocols. This level of understanding may be especially important for someone monitoring contractor testing or trying to isolate faults during system testing, as well as for someone evaluating a testing tool. To provide a sufficient level of understanding, there is a technical discussion about the fields of an encoded message.

TRAINING CLASS OUTLINES

1. Testing from the NTCIP Perspective

- 1.1. Introduction
 - 1.1.1 Conformance testing versus compliance testing.
 - 1.1.2 Conformance to NTCIP standards.
 - 1.1.3 Compliance to TxDOT specifications.
- 1.2. Background Information
 - 1.2.1 NTCIP standards promote interoperability, interchangeability, and compatibility. Interoperable is desirable so that system components from different vendors can work together. Interchangeable is desirable so that there is no loss in functionality when replacing system components with similar components from different vendors. Compatible is important so that system components can share a common communications infrastructure.
 - 1.2.1.1 Data Dictionaries define the words, and there is one dictionary for each field device.
 - 1.2.1.1.1 Objects / Data elements define parameters, controls, and status.
 - 1.2.1.1.2 Conformance Groups are collections of objects that together perform some specific function or task.
 - 1.2.1.1.3 Management Information Base (MIB) is a collection of objects related to field device. A MIB can come from multiple dictionaries.
 - 1.2.1.1.4 Protocol/Profile Requirements List (PRL) is a checklist of object support.
 - 1.2.1.1.4.1 PRL Object Types indicate what type of tests may apply to the objects.
 - 1.2.1.1.5 Requirements Traceability Matrices provide a mapping between user needs/requirements and objects that address them.
 - 1.2.1.2 Communications Protocols define the rules for combining the words and transmitting them over the media.

- 1.2.1.3 Application-level communications protocols handle interface between end-application (e.g., Signal Controller) and transportlevel protocols.
 - 1.2.1.3.1 Simple Network Management Protocol (SNMP) commands are Get, Set, GetNext, and Trap and it uses Basic Encoding Rules (BER) encoding rules. The NTCIP Guide has several examples of SNMP encoded messages.
 - 1.2.1.3.2 Simple Transportation Message Protocol (STMP) commands are Get, Set, and GetNext but it uses Octet Encoding Rules (OER) encoding rules and is limited to 13 predefined messages. The NTCIP Guide has an example of STMP encoded messages.
 - 1.2.1.3.3 Simple Fixed Management Protocol (SFMP) is the same as SNMP but Object Identifiers (OIDs) use a different node as a reference.
 - 1.2.1.3.4 File Transfer Protocol (FTP) is the same as that used on the Internet.
 - 1.2.1.3.5 Trivial File Transfer Protocol (TFTP) is the same as FTP but without all the directory commands and other features.
- 1.2.1.4 Transport-level communications protocols handle end-end connections and transfers as well as routing through networks.
 - 1.2.1.4.1 Transportation Transport (T2) Profile is for nonnetworked environments where throughput is a concern.
 - 1.2.1.4.2 User Datagram Protocol (UDP) is for a networked environment with best effort delivery.
 - 1.2.1.4.3 Transport Control Protocol (TCP) is for a networked environment with guaranteed delivery.
 - 1.2.1.4.4 Internet Protocol (IP) is for a networked environment and handles routing.
- 1.2.1.5 Subnetwork level protocols handle point-to-point connections and deal with issues related to putting information on media and errors.

- 1.2.1.5.1 Point to Multi-Point Protocol (PMPP) uses rules that are similar to the one that people use to dial into an internet service provider like AOL but supports party lining. There are separate standards for RS232 and FSK modems.
- 1.2.1.5.2 Point-to-Point Protocol (PPP) is the one that people use to dial into an internet service provider like AOL. PPP specifies the Challenge Handshake Authentication Protocol (CHAP) for authentication.
- 1.2.1.5.3 Ethernet Protocol is the same one used on office computers. It provides high throughput and robustness.
- 1.2.1.5.4 The NTCIP Guide has several examples of message encoding over various protocols.
- 1.2.1.6 NTCIP standards do not define a standard method of encrypting information. User access "passwords" are visible on the wire.
- 1.2.2 Other Standards
 - 1.2.2.1 NTCIP 8007 Testing and Conformity Assessment Documentation within NTCIP Standards Publications (NTCIP 8007-TEST) prescribes format of test procedures using natural language to describe test steps. The language includes basic keywords.
 - 1.2.2.2 NTCIP 9012 Testing and Conformity Assessment User Guide for NTCIP Field Devices and Center-to-Field Communications (NTCIP 9012-TG) provides general testing information in the context of ITS.
- 1.2.3 Implementations
 - 1.2.3.1 An implementation consists of a combination of four levels of standards: information, application, transport, and subnetwork.
 - 1.2.3.2 The MIB that defines an implementation comes primarily from the information-level data dictionary standards but can include objects related to communications level application, transport, and subnetwork standards.
- 1.3. Testing Terminology and Techniques
 - 1.3.1 Device Under Test (DUT) is the implementation that is undergoing the test.

- 1.3.2 Positive Testing exercises a DUT in a manner that is consistent with its normal operating conditions.
- 1.3.3 Negative Testing (a.k.a. Error Seeding) exercises a DUT in abnormal operating conditions such as out-of-range variables, input errors, and fault conditions.
- 1.3.4 Sampling is a technique that tests only a portion of the units with the assumption that the others will perform in a similar manner.
- 1.3.5 Regression is the retesting of a previously tested program following modification to ensure that faults have not been introduced.
- 1.3.6 Black-box Testing is based on an analysis of the specification of the component without reference to its internal workings as in White-box Testing.
- 1.3.7 Boundary Value Analysis is a selection of test values that surround the "boundaries" of a parameter's input range. Choices often include maximum, minimum, and trivial values.
- 1.3.8 Stress Testing subjects a system to incorrect, abnormal, or unrealistic inputs or conditions with the intention of producing a failure. It looks at testing at or beyond the limits of its specified requirements.
- 1.3.9 Validation and Verification (V&V) Verification is testing that determines whether an implementation is built correctly whereas validation testing checks whether the correct implementation was built.
- 1.4. Interpreting Results
 - 1.4.1 NTCIP PRLs standards provide Yes / No for each object or function and do not provide any additional guidance.
 - 1.4.2 NTCIP test procedures use a Pass / Fail method and do not provide any additional guidance other than a test case basis.
 - 1.4.3 Battelle's suggestion was to evaluate "critical" functions.
 - 1.4.4 Non-TxDOT Test Procedures
 - 1.4.4.1 Some Enterprise test procedures use a point scale with a minimum passing value.
 - 1.4.4.2 SimpleTester[™] uses a scale of 1 to 4 and scores on a test case basis.

- 1.4.5 Proposed TxDOT Test Procedures use NTCIP PRLs and test procedure forms.
 - 1.4.5.1 User determines what constitutes whether DUT is suitable.
- 1.4.6 Contractor Testing is unknown but likely varies by contractor.

2. Testing from the TxDOT Perspective

- 2.1. Risk Management
 - 2.1.1 Do you have to try all possible values in an object acceptable range of values?
 - 2.1.2 Do you have to check all instances of an object if there are 16 duplicates of the same thing?
 - 2.1.3 If a DUT supports all the objects in the specifications but does not support some object required by NTCIP, is it still acceptable to use?
 - 2.1.4 If a DUT supports all the values of an object required for a project but fails on some values that are called for in the specifications, is it still acceptable to use?
 - 2.1.5 Do all possible permutations and combinations need to be tested? For example, in a traffic signal controller, is a detector call for service on another phase entered when the current phase is green, yellow, and red? Does the call register when a traffic signal controller is in red rest?
- 2.2. What to Test
 - 2.2.1 Ensure that all required objects are readable.
 - 2.2.2 Check whether value of objects that represent some limit or number of instances meets or exceeds a required value.
 - 2.2.3 Ensure that parameter objects accept the required range of values.
 - 2.2.3.1 Perform boundary analysis with positive and negative range.
 - 2.2.3.2 Perform sampling of mid-range values.
 - 2.2.3.3 Perform sampling of multiple instance parameters.
 - 2.2.4 Ensure that control objects can be set to the required range.
 - 2.2.4.1 Goal is to test 100% of all values but tempered with realistic permutations and combinations.
 - 2.2.5 Verify that all status objects return the appropriate value when parameter and control objects are set accordingly.
 - 2.2.5.1 Goal is to check 100% of all values by creating scenarios that produce status values.
 - 2.2.6 Perform regression testing when software changes.

2.3. Testing Tools

- 2.3.1 Active tools simulate the operation of management application.
 - 2.3.1.1 Tools range from device-specific, canned testers to fully customizable, general-purpose tools.
 - 2.3.1.2 Cost runs from free to mid four figures.
 - 2.3.1.3 One should use caution in that it is best to verify that a tool does what it says it does.
- 2.3.2 Emulator tools act like field devices.
 - 2.3.2.1 Tools range from device-specific, canned emulators to generalpurpose that can be customized to simulate the entire system consisting of many different types of field devices.
- 2.3.3 Instrumentation can monitor and analyze information exchanges.
 - 2.3.3.1 Instrumentation can act as arbitrator when using other tools and during systems testing. Project engineer can use it to verify testing by contractors.
- 2.3.4 High-ended active and emulator tools have the ability to run user-defined scripts (programs) that define tool operation.
 - 2.3.4.1 Scripts are generally proprietary but one set of tools uses the Tool Command Language (TCL), a common language in many testing and simulation applications.
 - 2.3.4.2 Some tools allow interface to other programs (e.g., hardware-inthe-loop).
 - 2.3.4.3 Once procedures are written and verified, the process of testing is greatly simplified.
- 2.4. Configuration Management
 - 2.4.1 Specify NTCIP standard version numbers.
 - 2.4.1.1 Mentioning "or latest version" can lead to interoperability issues.
 - 2.4.1.2 Management application and field devices should support same version.
 - 2.4.2 Configuration management consists of identification, change control, configuration auditing, and reporting.

- 2.4.2.1 Identification involves enumerating what is currently being used to establish baseline and describing new versions as to applicability and what is changed.
- 2.4.2.2 Change control involves understanding the impact on performance reliability and compatibility.
- 2.4.2.3 Configuration auditing involves identifying what components will need updating.
- 2.4.2.4 Reporting involves making sure that everyone involved is kept informed.

TRAINING CLASS EVALUATION FORM

NTCIP Testing Training Course Evaluation Form

Locatio	on:			
Date:				
Your A	Agency:			
Your F	Position:			
Course	Content			
1.	Did the course meet your expectations? Comments:	Yes 1	Somewhat 2	No 3
2.	Was the material presented at the correct level of difficulty? Comments:	1	2	3
3.	Were the presentation and guidebook appropriately geared to providing you the information you needed? Comments:	1	2	3
4.	Do you feel the time spent on this course was beneficial? Comments:	1	2	3
Genera	l Observations			

- 5. What did you like most about the course?
- 6. What did you like least about the course?

- 7. What can we do to improve this course in the future?
- 8. Do you have any other suggestions or comments?

Thank you for taking the time to complete this course evaluation form. Please make sure the course instructor receives it before you leave.

CHAPTER 6: RECOMMENDATIONS

TESTING FRAMEWORK

The following are additional researcher's recommendations on defining a framework for the testing of conformance to NTCIP and integrating it into the current TxDOT testing program.

- 1. Review the detector operation requirements for detectors 1 and 5 with respect to the fourphase diamond operation that appears in DMS 11170-TSC. When the opposing through movement is green, a call for service on these detectors does not register.
- Reorganize special specifications titled National Transportation Communications for ITS Protocol for Field Equipment so that they distinctly identify information, application, transport, and subnetwork-level profiles and protocols requirements. Along with this, consider whether the application, transport, and subnetwork-level profiles and protocols requirements can be stand-alone documents.
- Add wording in any specification that relates to NTCIP to address application, transport, and subnetwork-level object conformance groups. Support for the object definitions could help when experiencing communications problems.
- 4. Consider whether any planning for test documentation should follow the IEEE Standard for Software Test Documentation. This would help in any transition to ISO certification.

FUTURE DEVELOPMENT

ELMS Test Procedures

The NTCIP 1213-ELMS standard has progressed to the stage where it includes a set of dialogs that illustrate the exchange of information between a management application and a field device (*31*). Text also describes how a management application would carry out the exchange of information. What it lacks, however, is a standard set of procedures to verify a correct implementation. There are no plans within the NTCIP development process to add test procedures to the standard. Even though TxDOT representatives made significant contributions to the development of the NTCIP ELMS standard, the TxDOT specifications for ELMS do not reference any NTCIP requirements.

From the findings of this research project, the process and steps to follow for the development of ELMS test procedures are in place. A research or implementation project that focuses on ELMS would provide templates for TxDOT documentation that specifies NTCIP requirements. A set of test procedures would create a 'conformant management station' as defined in the NTCIP-1213 ELMS. A conformant management station would provide TxDOT personnel with the means to test for conformance to the NTCIP standards and test for compliance to the TxDOT specifications.

Generic Database

In the course of developing the four-phase diamond detector operations test procedures for traffic signal controllers, it was necessary to load numerous timing parameters and set various controls in order for the controller to perform as expected. In some cases, the loading and setting involved four parameters for each of 16 phases. When considering other test procedures for traffic signal controllers, there may be hundreds of parameters to be set in order to create suitable test conditions. Other devices have similar complexity.

A test procedure can specify steps to retrieve and save the current parameter values, download appropriate values for the test, and then restore the original values when the test is complete. An alternate approach would be to develop a generic database for the purposes of testing and configuring ITS field devices. The design of the database could use the selfdescribing information in a Management Information Base that is part of every device standard. A MIB names the parameters, controls, and status information of a device, defines data types and constraints for data entries, provides a description of the data, and describes the structure or organization of data. Organizing this information as a spreadsheet would provide an easy method of entering test values and a convenient way of changing the parameters of a device in a test procedure. This database would create a reusable utility and eliminate the need to hard code test values in a test script.

Personnel at the Florida Traffic Engineering Research Lab (TERL) and at the Idaho National Institute for Advanced Transportation Technology (NIATT) have expressed a need for database utility in their testing activities. One of the AASHTO manufacturer representatives on the NTCIP oversight committee believes that a generic database would also help companies that do have device specific database support in their management application software. Any MIB-

90
based database would never have the ease of use, refined user interface, or understanding of the relationships in the data that manufacturers build into their management application software. Research into the definition of a generic MIB-based database has the potential to simplify the task of defining data sets and providing access to device information when specific management application software is not available.

REFERENCES

- Special Specification 6025 CCTV Field Equipment. Published by TxDOT. <u>ftp://ftp.dot.state.tx.us/pub/txdot-info/cmd/cserve/specs/2004/spec/ss6025.pdf</u>. Accessed June 21, 2006.
- DMS-11170, Fully Actuated, Solid-State Traffic Signal Controller Assembly, Departmental Material Specifications 7-115, Section 19. Published by TxDOT. <u>http://manuals.dot.state.tx.us/dynaweb/colmates/dms/@ebt-link;?target=idmatch(s070019)</u>. Accessed July 29, 2005.
- *3.* IEEE Std 829-1998 IEEE Standard for Software Test Documentation, Institute of Electrical and Electronics Engineers, New York, New York, 1998.
- 4. TxDOT Specifications, <u>http://www.dot.state.tx.us/business/specifications.htm</u>. Accessed August 21, 2006.
- 5. NTCIP 1203 Object Definitions for Dynamic Message Signs (DMS), A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=1203</u>. Accessed August 17, 2006.
- 6. NTCIP 1205 Object Definitions for Closed Circuit Television (CCTV) Camera Control, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=1205</u>. Accessed August 17, 2006.
- 7. NTCIP 1201 Global Object Definitions, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=1201</u>. Accessed August 17, 2006.
- NTCIP 2301 Simple Transportation Management Framework Application Profile, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=2301</u>. Accessed July 25, 2005.
- 9. NTCIP 2201 Transportation Transport Profile, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar_d=2101</u>. Accessed August 17, 2006.
- 10. NTCIP 2101 Point to Multi-Point Protocol Using RS-232 Subnetwork Profile, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=2101</u>. Accessed August 17, 2006.

- *11.* Special Specification 6026 National Transportation Communications for ITS Protocol for Dynamic Message Signs. Published by TxDOT. <u>ftp://ftp.dot.state.tx.us/pub/txdot-info/cmd/cserve/specs/2004/spec/ss6026.pdf</u>. Accessed August 17, 2006.
- *12.* NTCIP 9001 The NTCIP Guide, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=9001</u>. Accessed September 12, 2006
- 13. NTCIP 2103 Point-to-Point Protocol over RS-232 Subnetwork Profile, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=2103</u>. Accessed August 17, 2006.
- 14. Rose, M., and K. McCloghrie, Management Information Base for Network Management of TCP/IP-based internets: MIB-II, RFC 1213, Performance Systems International, Hughes LAN Systems, March 1991.
- Test Procedures for NTCIP-conformant Closed Circuit Television (CCTV) Camera Controllers, Enterprise Consortium. <u>http://enterprise.prog.org/</u> (document no longer available). Accessed December 2002.
- 16. NTCIP 1202 Object Definitions for Actuated Traffic Signal Controller Units, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=1202</u>. Accessed August 17, 2006.
- 17. TS 2-2003 Traffic Controller Assemblies with NTCIP Requirements, National Electrical Manufacturers Association, Rosslyn, Virginia.
- 18. NTCIP 2104 Ethernet Subnetwork Profile, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=2104</u>. Accessed June 27, 2006.
- 19. Travel Time Messaging on Dynamic Message Signs Portland, OR. Published by the Federal Highway Administration. <u>http://ops.fhwa.dot.gov/publications/travel_time_study/portland/portland_ttm.htm</u>. Accessed July 17, 2006.
- 20. The City of Palo Alto Uses GarrettCom Ethernet Products in Its Advanced Transportation Management System (ATMS), A GarrettCom Application Note. Published by GarrettCom, Inc. <u>http://www.garrettcom.com/techsupport/appnotes/paloalto_appnote.pdf</u>. Accessed July 17, 2006.

- 21. ISO 9000 and ISO 14000 in brief, ISO 9000. <u>http://www.iso.org/iso/en/iso9000-14000/understand/inbrief.htm</u>. Accessed August 16, 2006.
- 22. NTCIP 8007 Testing and Conformity Assessment Documentation within NTCIP Standards Publications, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=8007</u>. Accessed August 17, 2006.
- 23. NTCIP 1103 Transportation Management Protocols, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=1103</u>. Accessed August 17, 2006.
- 24. NTCIP 1204 Environmental Sensor Station Interface Standard, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standards/default.asp?documents=yes&qreport=n</u>
- 25. NTCIP 1206 Object Definitions for Data Collection and Monitoring (DCM) Device, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=1206</u>. Accessed August 17, 2006.
- 26. NTCIP 1207 Object Definitions for Ramp Meter Control (RMC) Units, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=1207</u>. Accessed August 17, 2006.
- 27. NTCIP 1208 Object Definitions for Closed Circuit Television (CCTV) Switching, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=1208</u>. Accessed August 17, 2006.
- 28. NTCIP 1209 Data Element Definitions for Transportation Sensor Systems, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=1209</u>. Accessed August 17, 2006.
- 29. NTCIP 1210 Field Management Stations Part 1: Object Definitions for Signal System Masters, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=1210</u>. Accessed August 17, 2006.
- *30.* NTCIP 1211 Object Definitions for Signal Control and Prioritization, A Joint Publication of AASHTO, ITE, and NEMA.

http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=1211. Accessed August 17, 2006.

- 31. NTCIP 1213 Objects Definitions for Electrical and Lighting Management Systems, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=1213</u>. Accessed August 17, 2006.
- *32.* Tex-1170-T, Sampling and Environmental Testing of Traffic Signal Controller Assemblies: Traffic Signal Controllers and Conflict Monitors. Published by TxDOT. <u>ftp://ftp.dot.state.tx.us/pub/txdot-info/cst/TMS/1100-T_series/pdfs/tsi1170.pdf</u>. Accessed July 29, 2005.
- 33. Special Specifications 6504 Testing, Training, Documentation and Warranty. Published by TxDOT.
 <u>ftp://ftp.dot.state.tx.us/pub/txdot-info/cmd/cserve/specs/1993/spec/es6504.pdf</u>. Accessed June 30, 2005.
- 34. NTCIP 9012 Testing and Conformity Assessment User Guide for NTCIP Field Devices and Center-to-Field Communications, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=9012</u>. Accessed August 31, 2006.

APPENDIX A SPECIAL SPECIFICATION FOR CCTV EQUIPMENT

The following is a modified version of the TxDOT 2004 Special Specification 6025 – CCTV Field Equipment (1). The modification to the specification consists of adding a reference to another special specification defining the details of the NTCIP requirements, updating wording to be consistent with requirements in the NTCIP standards, and removing requirements that would be in conflict with or superseded by NTCIP requirements.

Italics highlight the additions to the document and strikethrough highlights the deleted wording.

SPECIAL SPECIFICATION 6025 (Modified for NTCIP Requirements) CCTV Field Equipment

1. Description. Furnish and install closed circuit television (CCTV) field equipment.

The following special specification is referenced in this specification: "National Transportation Communications for ITS Protocol for CCTV Equipment"

2. Materials. Provide new, corrosion resistant materials in accordance with the details shown on the plans and this item.

Provide CCTV field equipment including, but not limited to, the following:

- Color video camera units.
- Camera lenses, filters, control circuits and accessories.
- Camera housings.
- Medium duty pan and tilt units.
- Camera control receivers.
- Video and camera control and power cable harnesses, connectors and coaxial cable.
- Equipment for accommodating presets.
- Source ID Generator.
- When shown on the plans, Local Control Panel.
- **A. Functional Requirements.** Provide CCTV Cameras in accordance with NTSC and EIA-170A. Conform the system limiting resolution to FCC regulations for broadcast signals. Provide clear, low-bloom and low-lag video pictures under all conditions from

bright sunlight to nighttime scene illumination of 0.1 ft.-candle (fc.). Maintain color quality by a continuous through the lens automatic white balance for color temperatures from 2850°K to greater than 5100°K with less than 10 IRE units unbalance.

Provide field equipment that operates in all weather conditions and able to withstand a wind load of 80 mph without permanent damage to mechanical and electrical equipment, unless otherwise shown on the plans.

Provide equipment from the same manufacturer at each field location.

B. Electrical and Mechanical Requirements.

- **1. Video Camera Unit.** Provide color video cameras of solid state design, and that meet the following requirements:
- Use Digital Signal Processing (DSP):
 - For digital zoom;
 - For Auto/Manual long-term integration (exposure) control, with built-in frame buffer;
 - For Auto-focus;
 - For built-in I.D. Generator, with white letters and black outline.
- **Image Pickup Device:** 1/4 in. single chip interline transfer solid state color matrix CCD microlens sensor
- **Pickup Device Blemishes:** When viewing a uniform white field, there must be no blemishes for any iris opening producing any signal level between 7.5 and 100 IRE.
- **Sensitivity:** Maintain full p-p video with 0.1 fc. 3200°K incandescent illumination on the image device face plate with AGC off.
- **Resolution:** > 350 lines vertical and > 460 lines horizontal, measured per EIA-170A Standard.
- **Over Exposure Protection:** The camera must not sustain any permanent damage when pointed directly at strong light sources, including the sun, for brief periods of time.
- **Encoded NTSC Video Signal Format**: EIA-170A Standard, video output 1 Volt p-p composite. Must have up to 16 dB AGC.
- **Output Impedance:** 75 Ohms \pm 5%.
- Aspect Ratio: 4:3.
- **Geometric Distortion**: Zero.
- Signal to Noise Ratio (AGC Off): 55 dB minimum (weighted at 4.5 MHz).
- Sensor with a minimum of 768(H) X 493(V) pixels.
- Lens must be integral to camera assembly.
- Electronic Shutter Speed: software selectable, remotely.

- **2.** Camera Lens. Provide an integral lens assembly for each camera with the following features:
- An f/1.6 or better glass multi-coated zoom lens. The lens must have variable focal length from 3.9 mm to 85.8 mm.
- Provide motorized iris control with manual override with each lens.

Provide a lens with capabilities for remote control of zoom, focus and iris operations. Provide mechanical or electrical means to protect the motors from overrunning in extreme positions. The lens and controller system must be capable of both auto iris, and remote manual iris operation. Iris must be "motorized", as opposed to "auto iris" type, for system control compatibility.

3. Camera Housing. Furnish and install an environmental resistant and tamperproof housing pressurized to 5 psi dry Nitrogen with Schrader purge fitting and 20 psi relief valve for each camera.

Except for the viewing window, construct the enclosure from 6061-T6 standard aluminum tubing with a wall thickness of 0.20 in. \pm 0.03 in. Label internal wiring properly. Use a gas-tight connector at the rear plate of the housing.

The internal humidity of the housing must be less than 10%, when sealed and pressurized. Securely place desiccant packs inside the housing to absorb any residual moisture and maintain internal humidity at 10% or less.

Provide a low pressure sensor in the camera to put a "low-pressure" annotation on the video signal through the internal I.D. generator.

Construct the viewing window in such a way that unrestricted camera views can be obtained at all camera and lens positions.

Provide a sun shield to shield the entire housing from direct sunlight and vertical rainfall. Construct it in such a way as to allow the free passage of air between the housing and the shield, but it must not form a "sail" to place an excessive load on the pan/tilt unit in high winds.

Provide with an internal 15 W. low temperature heater with its own thermostat control in each housing.

Provide lightning protection as shown on the plans in each housing.

4. **Pan-Tilt Unit.** Furnish and install a medium duty, anodized aluminum weatherproof pan-and-tilt unit at each camera site on top of the camera pole. Provide a mounting plate to install the unit on the pole. Design the mounting for the camera housing and the pan-and-tilt unit to withstand the wind loading specified in Section 2.A.

Provide a unit with vertical movement of $+40^{\circ}$ to -90° and horizontal movement of 360° full, contiguous rotation movement. Tilt speed must be 20° per sec. and the pan speed must be up to 100° per sec. Provide a unit that is capable of simultaneous pan-and-tilt movements.

Provide a unit with a load rating compatible with that of the camera housing, camera and cabling under wind conditions specified in Section 2.A. and acceleration/deceleration conditions specified. Provide analyses of the loading on the pan-and-tilt assembly based on the above criteria.

Use Stepper motors.

Provide pan-and-tilt units that have seals and gaskets to protect the motors, gears, and cables. Provide seals and gaskets that are resistant to ozone, ultraviolet radiation, and other pollutants inherent to local environmental conditions.

- **5.** Local Control Panel. Provide Local Control Panel that meet the following specific requirements without use of a laptop:
- Pan Left.
- Pan Right.
- Tilt Up.
- Tilt Down.
- Zoom In.
- Zoom Out.
- Focus Near.
- Focus Far.
- Manual and Auto Iris control.
- Iris Open.
- Iris Close.
- Pan/Tilt Position preset.
- Camera Power (Latching).
- Remote white balance control.
- Auto and Manual white balance control.
- Zoom and focus position preset.
- 6. Control Receivers. Mount the camera control receiver inside the camera unit. It must execute camera and lens functions and must also forward communication of commands for the pan/tilt functions to the pan/tilt control receiver. Mount the pan/tilt control receiver inside the pan/tilt unit. Provide camera and pan/tilt functions that are operable via RS-422 serial communications.

Provide control receivers that receive the command data from the camera controller and decode the digital command data signals transmitted through the communication transmission interface, perform error checking and act on valid data to drive the pan/tilt unit and the camera controls. Detail the communications transmission interface on the plans. Provide control receivers that are fully compatible with the existing camera controller shown on the plans. Provide control receivers that meet the following specific requirements:

- **Camera remote control functions**: Provide units with, as a minimum, control and drive circuits for the following functions:
 - **DSP Functions**: Zoom, Long-Term Exposure, Auto-Focus, Auto/Manual focus Control, I.D. Generator Operation, and Alarm function Control.
 - Pan/Tilt Position preset.
 - Pan Left.
 - Pan Right.
 - Tilt Up.
 - Tilt Down.
 - Zoom and focus position preset.
 - Zoom In.
 - Zoom Out.
 - Focus Near.
 - Focus Far.
 - Manual and Auto Iris control.
 - Iris Open.
 - Iris Close.
 - Camera Power (Latching).
 - Remote white balance control.
 - Auto and Manual white balance control.
 - One auxiliary output (unless specified otherwise in the plans).
- **Controller Address**: Provide each unit with a unique programmable address. Provide units that respond to the central command if and only if they are addressed.
- **Power Supplies**: Provide power supplies required to operate the camera, pan/tilt, and lens movements and include them with the housing, camera control receiver, and pan/tilt unit.
- Communications Interface: Provide a camera control receiver that interfaces to the communications backbone through an *RS* EIA-232 Serial C/D-port and shall be in accordance with special specification National Transportation Communications for ITS Protocol for CCTV Equipment. When indicated on the plans, provide communications signals, data exchange protocol and timing that is compatible with the communications equipment and with the existing master controller in the satellite building. Use a minimum 9600 Baud data rate. Data must be sent asynchronously as either 8 bit with no parity.-or 7 bit with parity. Each block of data must include a camera identifier and be accompanied by a checksum calculated on the entire block. Blocks with a bad checksum must be NAKed. Block with a good checksum must be ACKed. If the field unit must transmit data to the control unit at the Satellite Building,

it must raise the RTS line and keep it raised until all data has been sent. Provide a field unit that will not transmit data unless the CTS line from the communications equipment is raised. Provide the camera control receiver connectors and harness to connect to the communications equipment interface. Supply complete hardware interface and protocol description to the Department as part of the required documentation.

Provide RS-232 to RS-422 external powered converter that is an integral part of the video communication junction box.

- **Power Input**: 115 VAC plus or minus 10%, 60 Hz \pm 3 Hz, 50 W. Maximum.
- **Connectors**: Provide and install connectors which are compatible with the communications equipment interface. Use Connectors for connections at the pan/tilt mechanism. Make connections through a pigtail with a connector on it coming out of the bottom center of the pan/tilt unit. Provide the connector on the pigtail that is an AMP type connector. Provide connections down to the pole to the transmission cables to this connector. Supply mating connectors. Provide connector pins and mating connectors that are plated to ensure good electrical connection and resist corrosion. Use pressure tight multi-conductor MS-type cable connectors for camera connections.
- 7. Source ID Generator. Provide the built-in I.D. Generator that inserts camera ID over each of the camera generated videos.

Submit a list of proposed camera identification text to the Engineer for approval before the ID is programmed.

Once programmed, the programmed ID must automatically be displayed with its associated video signal.

Provide the source ID generator that will automatically "pass through" video in case of equipment failure.

When indicated on the plans, provide the source ID generator that is compatible with the existing camera controller shown on the plans.

- 8. Video Communication Junction Box. Install the video communication junction box in the CCTV equipment cabinet or in the surveillance cabinet, as shown on the plan and as directed by the Engineer. Provide the video communication junction box that contains the lightning protection devices for data, power, and video. The junction box must be grounded very well to the earth ground. Provide the junction box that has connectors for inputs and outputs for data, power, and video. Make testing and connections to communication devices through these external connectors.
- **9.** Surge Protection. Provide the camera installation that meets the following requirements:
- Pole mounting adapter -- Electrically bonded to pole.
- Pan/tilt mechanism -- Electrically bonded to adapter.

- Camera housing -- Electrically bonded to pan/tilt unit.
- **10.** Power and Control Cable Surge Protector. Protect each power conductor and each control conductor (including return conductors) by the appropriate surge protector. House the protective devices in each of the surveillance cabinets.
- 11. Power Requirements. Provide CCTV field equipment that meets its specified requirements when the input power is $115 \text{ VAC} \pm 10\%$, $60 \text{ Hz} \pm 3 \text{ Hz}$. The maximum power required must not exceed 350 W.

Provide equipment operations that are not affected by the transient voltages, surges and sags normally experienced on commercial power lines. Check the local power service to determine if any special design is needed for the equipment. The extra cost, if required, must be included in the bid of this item.

- **12. Primary Input Power Interruption.** Provide CCTV field equipment that meets the requirements in Section 2.1.4. "Power Interruption" of the NEMA Standard TS2 for Traffic Control System.
- **13. Power Service Transients.** Provide CCTV field equipment that meets the requirements of Section 2.1.6., "Transients, Power Service" of the NEMA Standard TS2.
- 14. Wiring. Provide wiring that meets the requirements of the National Electric Code. Provide wires that are cut to proper length before assembly. Do not doubled-back wire to take up slack. Lace wires neatly into cable with nylon lacing or plastic straps. Secure cables with clamps. Provide service loops at connections.

Provide coaxial cable between the camera and the communications equipment interface that is of the RG-59 type with a stranded center conductor and 100% shield coverage. Provide coaxial cable that has a cellular polyethylene dielectric.

- **15. Transient Suppression.** Provide DC relays, solenoids and holding coils that have diodes or other protective devices across the coils for transient suppression.
- **16. Power Service Protection.** Provide equipment that contains readily accessible, manually resettable or replaceable circuit protection devices (such as circuit breakers or fuses) for equipment and power source protection.

Provide and size circuit breakers or fuses such that no wire, component, connector, PC board or assembly must be subjected to sustained current in excess of their respective design limits upon the failure of any single circuit element or wiring.

- **17. Fail Safe Provision.** Provide equipment that is designed such that the failures of the equipment must not cause the failure of any other unit of equipment.
- **18.** Modular Design. Provide CCTV field equipment that is modular in design to allow major portions to be readily replaced in the field. Identify modules and

assemblies clearly with name, model number, serial number and any other pertinent information required to facilitate equipment maintenance.

19. Connectors and Harnesses. Provide external connections made by means of connectors. Provide connectors that are keyed to preclude improper hookups. Color code and/or appropriately mark wires to and from the connectors.

Provide connecting harnesses of appropriate length and terminated with matching connectors for interconnection with the communications system equipment.

Provide pins and mating connectors that are plated to improve conductivity and resist corrosion. Cover connectors utilizing solder type connections by a piece of heat shrink tubing securely shrunk to insure that it protects the connection.

- **C.** Environmental Design Requirements. Provide equipment that meets its specified requirements during and after subjecting to any combination of the following conditions.
 - Ambient temperature range of 0°F to 140°F.
 - Temperature shock not to exceed 30°F per hour during which the relative humidity must not exceed 95%.
 - Relative humidity range not to exceed 95% over the temperature range of 40° F to 110° F.
 - Moisture condensation on exterior surfaces caused by temperature changes.

Provide camera and environmental housing assemblies that perform to stated specifications over an ambient temperature range of -35° F to $+130^{\circ}$ F and a humidity range of 0% to 100 % condensing. The camera must operate without sustaining damage over temperature range of -35° F to 140° F.

3. Construction Methods.

A. General. Provide equipment that utilizes the latest available techniques for design and construction with a minimum number of parts, subassemblies, circuits, cards, and modules to maximize standardization and commonality.

Design the equipment for ease of maintenance. Provide component parts that are readily accessible for inspection and maintenance. Provide test points that are for checking essential voltages and waveforms.

- **B.** Electronic Components. Provide electronic components in accordance with Special Specification, "Electronic Components".
- **C.** Mechanical Components. Provide external screws, nuts and locking washers that are stainless steel; no self-tapping screws will be used. Provide parts made of corrosion resistant material, such as plastic, stainless steel, anodized aluminum or brass. Protect materials from fungus growth and moisture deterioration. Separate dissimilar metals by an inert dielectric material.

- **4. Testing.** Perform testing in accordance with Article 2, Special Specification, "Testing, Training, Documentation, Final Acceptance, and Warranty".
- **5. Training.** Provide training in accordance with Article 3, Special Specification, "Testing, Training, Documentation, Final Acceptance, and Warranty".
- **6. Documentation.** Provide documentation in accordance with Article 4, Special Specification, "Testing, Training, Documentation, Final Acceptance, and Warranty".
- 7. Warranty. Provide a warranty in accordance with Article 6, Special Specification, "Testing, Training, Documentation, Final Acceptance, and Warranty".
- 8. Measurement. This Item will be measured as each unit furnished, installed, and tested.
- **9. Payment.** The work performed and materials furnished in accordance with this Item and measured as provided under "Measurement" will be paid for at the unit price bid for "CCTV Field Equipment". This price is for equipment, cables and connectors; documentation and testing; and labor, materials, warranty, training and incidentals.

REFERENCES FOR APPENDIX A

 Special Specifications 6025 – CCTV Field Equipment. Published by TxDOT. <u>ftp://ftp.dot.state.tx.us/pub/txdot-info/cmd/cserve/specs/2004/spec/ss6025.pdf</u>. Accessed June 21, 2006.

APPENDIX B SPECIAL SPECIFICATION - NTCIP FOR CCTV EQUIPMENT

The following is a template for the wording for a new TxDOT 2004 Specification that defines NTCIP for CCTV equipment. The organization and content of this new specification is modeled after TxDOT Special Specification 6026 (1). Chapter 2 of this report provides an explanation of each item and the reasoning behind a number of choices. Also, please note the specification of the transport and subnetwork communications protocols are "direct connect" and based upon NTCIP 2101-PMPP/RS232 and NTCIP 2201-T2 (2,3). These protocols and standards reference nonnetworked serial communications. Networked communications protocols such as NTCIP 2104-Ethernet and NTCIP 2202-ITP are viable alternatives (4,5).

SPECIAL SPECIFICATION

XXXX

National Transportation Communications for ITS Protocol for CCTV Equipment

- **1. Description.** Provide Closed Circuit Television (CCTV) software that complies with the National Transportation Communications for ITS Protocol (NTCIP).
- **2. Requirements.** Ensure software complies with the NTCIP Standards when installed. Ensure software complies with the relevant current NTCIP standards, including associated amendments. The term "software" includes both software and firmware.

[Official printed copies of the NTCIP Joint Standards Publications referenced in this specification may be purchased from Global Engineering Documents, phone 1-800-854-7179, or <u>http://www.global.ihs.com</u>. They are also freely available in Adobe Acrobat PDF format at <u>http://www.ntcip.org/library/documents/</u>.]

- A. Ensure software complies with NTCIP 2101 Point-to-Multi-Point Protocol Using RS-232 Subnetwork Profile (2001) (direct connect).
- **B.** Ensure software complies with NTCIP 2201 Transportation Transport Profile (v01.14), and shall meet the requirements of parsing method 1 and encapsulation method 1.
- C. Ensure software complies with NTCIP 2301 Simple Transportation Management Framework – Application Profile (2001), and shall meet the requirements of Conformance Level 1 (SNMP).

- D. Ensure software implements all mandatory objects of the mandatory and optional conformance groups as defined in NTCIP 1205 Object Definitions for Closed Circuit Television (CCTV) Camera Control (v01.08a) as follows:
 - CCTV Configuration with support for clauses 3.2, 3.3, and 3.11
 - Extended Functions with support for clauses 3.6, 3.7, and 3.9
 - Motion Control with support for clauses 3.4 and 3.5
 - Configuration
 - Security
- E. Ensure software implements the following optional objects defined in NTCIP 1205 Object Definitions for Closed Circuit Television (CCTV) Camera Control (v01.08a) as follows:
 - positionQueryFocus
 - positionQueryIris
- **F.** Ensure software implements all mandatory objects of the mandatory conformance groups as defined in NTCIP 2301 Simple Transportation Management Framework Application Profile (2001):
 - System Group
 - SNMP Group
 - SNMP Configuration
- **G.** Ensure software implements the mandatory objects of the mandatory conformance groups as defined in NTCIP 2101 Point-to-Multi-Point Protocol Using RS-232 Subnetwork Profile (2001):
 - HDLC Group
 - RS232 Asynchronous Group
 - HDLC Group Address Group
- **H.** Ensure that objects that are required to support this NTCIP requirement support all values within its standardized range. Standardization range is defined by a size, range, or enumerated listing indicated in the object's SYNTAX field and/or through the descriptive text in the object's description field of the relevant standard.

OBJECT	MINIMUM PROJECT REQUIREMENTS
NTCIP 1205	
zoneMaximum	16
labelMaximum	16
labelColor	7 and 16
rangeMaximumPreset	
rangePanLeftLimit	355 degrees (SS 6860) ¹
rangePanRightLimit	
rangePanHomePosition	
rangeTiltUpLimit	+90 degrees (SS 6287, SS 6860) +40 degrees (SS 6225) +20 degrees (SS 6973)
rangeTiltDownLimit	-90 degrees (SS 6287, SS 6860, SS 6225) -110 degrees (SS 6973)
rangeZoomLimit	
rangeFocusLimit	
rangeIrisLimit	
rangeMinimumPanStepAngle	
rangeMinimumTiltStepAngle	
systemCameraEquipped	
systemLensEquipped zoneCameraEquipped	
menuControl	19

The following table provides the current listing of known variances for this project.

¹ Researcher's Note: The SS XXXX numbers refer to TxDOT special specification numbers that define different values for the parameter.

Tilt Speed	3-4 degrees per second (SS 6287, SS 6142, SS 6860) 20 degrees per second (SS 6225) 7 degrees per second (SS 6973)
Pan Speed	5-6 degrees per second (SS 6287, SS 6142, SS 6860) 100 degrees per second (SS 6225) 10 degrees per second (SS 6973)

OBJECT	MINIMUM PROJECT REQUIREMENTS
NTCIP 1201	
communityNamesMax	3
maxGroupAddresses	1

- I. Hardware Limitations: Ensure that a "noSuchName" SNMP error (OID not supported) will be returned by firmware for required objects that cannot be implemented due to hardware limitations.
- **J. Documentation:** Ensure software is supplied with full documentation, including a CD-ROM containing ASCII versions of the following Management Information Base (MIB) files in Abstract Syntax Notation 1 (ASN.1) format:
 - The relevant version of each official standard MIB Module referenced by the device functionality.
 - A manufacturer-specific version of the official Standard MIB Module with the supported range indicated in ASN.1 format in the SYNTAX field of the associated OBJECT TYPE macro for devices that do not support the full range of any object within a Standard MIB Module. Ensure the file name is identical to the Standard MIB Module, except having the extension ".man".
 - A MIB containing any other objects supported by the device.

Allow unrestricted use of this documentation by any authorized party for systems integration purposes, regardless of what parties are involved in the systems integration effort.

Provide documentation of any procedural implementation details for function(s) shown in the following table, wherever multiple objects are required to implement a feature required by the CCTV Specification. The following table enumerates which objects are used and any special procedures or required sequences to implementing that feature of function. Submit this table for approval prior to implementation.

Function	Objects	Procedures to Implementation
Example: Function X	Object T	Get object T then send objects Y
	Object Y	and Z if T>0.
	Object Z	
Pan		
Tilt		
Zoom		
Focus		
Iris		
Presets		
Camera Feature Control		
Camera Zones and Labels		
Camera Timeouts		
Camera Range		
Camera Alarms		
NTCIP Security		
Retrieve Module Table		

3. Testing and Verification. Demonstrate conformance to the applicable sections of NTCIP using approved test software. Ensure that conformance testing is performed by a qualified independent testing firm and witnessed and certified by a Professional Engineer. Executable code and documentation for the test procedures shall be available upon request from:

Mr. Carlos A. Lopez, P.E. Director, Traffic Operations Division Texas Department of Transportation 125 E. 11th Street Austin, Texas 78701-2483

The department reserves the right to have a representative witness all conformance tests. Test results will be compared to the requirements specified in this item. Failure to meet these requirements will be counted as a defect, and the software and associated hardware will be rejected. Final inspection and acceptance of software and associated hardware will be made after installation and performance testing at the designated locations as shown on the plans, unless otherwise directed.

4. Measurement and Payment. No direct measurement or payment will be made for the work performed and materials furnished in order to provide Closed Circuit Television (CCTV) software that complies with the National Transportation Communications for ITS Protocol (NTCIP) in accordance with this specification.

REFERENCES FOR APPENDIX B

- Special Specifications 6026 National Transportation Communications for ITS Protocol for Dynamic Message Signs. Published by TxDOT. <u>ftp://ftp.dot.state.tx.us/pub/txdotinfo/cmd/cserve/specs/2004/spec/ss6026.pdf</u>. Accessed June 21, 2006.
- NTCIP 2101– Point to Multi-Point Protocol Using RS-232 Subnetwork Profile, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=2101</u>. Accessed June 27, 2006.
- 3. NTCIP 2201 Transportation Transport Profile, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=2201</u>. Accessed June 27, 2006.
- 4. NTCIP 2104 Ethernet Subnetwork Profile, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=2104</u>. Accessed July 25, 2005.
- 5. NTCIP 2202 Internet (TCP/IP and UDP/IP) Transport Profile, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=2202</u>. Accessed July 25, 2005.

APPENDIX C: CCTV TEST PROCEDURES

INTRODUCTION

This appendix contains a set of test procedures for NTCIP conformant CCTV field equipment. Except for the prequalification test, the procedures are organized alphabetically by function and do not reflect any preferred sequence of execution.

TEST CASE SUMMARY

A summary of the test procedures and test cases for the features and/or functional areas derived from NTCIP 1205-CCTV is provided in Table C-1 (1). Two procedures, Global Configuration and Security, have a reference in NTCIP 1205-CCTV but the object definitions appear in NTCIP 1201-GLO (1,2). The majority of these test cases are derived from the Enterprise test procedures but are formatted to conform to NTCIP 8007-TEST (3,4). Some of the test procedures and/or test cases may not be applicable to CCTV field equipment that complies with TxDOT specifications because there is no TxDOT requirement for the feature or function. The test procedures do not address all the functional requirements of the TxDOT specification in that NTCIP does not support some of the TxDOT features or functions. Some test procedures also address features or functions that do not have a TxDOT specification requirement.

CCTV Test Cases					
ID	Title	Description			
	Pr	equalification			
TC001	CCTV PRL Information	This procedure retrieves minimum project requirements and maximum values, checks for whether the required objects are implemented, and performs a sampling of the supported values.			
		Alarms			
TC001	Cabinet Alarm	Tests cabinet open alarm and label associated with it			
TC002	Enclosure Alarm	Tests enclosure alarm and label associated with it			
TC003	Video Loss Alarm	Tests video loss alarm and label associated with it			
TC004	Temperature Alarm	Tests temperature alarm and label, thresholds, and current value associated with it			
TC005	Pressure Alarm	Tests the pressure alarm and label, thresholds, and current value associated with it			
TC006	Local Remote Alarm	Tests the local-remote alarm and label associated with it			
TC007	Washer Fluid Alarm	Tests washer fluid alarm and label, thresholds, and current value associated with it			
	Configuration				
TC001	Identify Device	Tests whether device under test (DUT) contains			
		version number			
TC002	Identify Preset Position Range	Ensures that device indicates that it supports the required number of preset positions			
TC003	Identify Pan Limits	Identifies and verifies the left and right panning limits and the home position of the device			
TC004	True North Offset	Ensures that the user can configure the true north			
TC005	Identify Tilt Limits	Ensures that the device indicates that it supports up and down tilting limits of the device			
TC006	Identify Zoom Limits	Ensures that the device indicates that it supports the required zoom limit			
TC007	Identify Focus Limits	Ensures that the device indicates that it supports the required focus limit			

Table C-1. CCTV Test Case Summary.

CCTV Test Cases			
ID	Title	Description	
TC008	Identify Iris Limits	Ensures that the device indicates that it supports the required iris limit	
TC009	Identify Pan-Tilt Step Angle Minimum	Ensures that the device indicates that it supports the pan and tilt step angle minimum	
TC010	Identify Zone Functions	Ensures that the device indicates whether it supports zones, zone labels, and control within a zone	
	Discrete	Input and Output	
TC001	Monitor Discrete Input	Verifies the state of discrete inputs and associated label	
TC002	Monitor Discrete Output	Verifies the state of discrete outputs and associated label	
		Features	
TC001	Get Availability of Equipment	Identifies and verifies the availability of attached equipment to the camera	
TC002	Control Camera Power	Enables and disables camera power while the user verifies	
TC003	Control Heater Power	Enables and disables heater while user verifies	
TC004	Control Wiper	Enables and disables wiper while user verifies	
TC005	Control Washer	Enables and disables washer while user verifies	
TC006	Control Blower	Enables and disables blower while user verifies	
		Focus	
TC001	Delta Focus Motion	Tests the delta focus motion of the camera by moving the camera with two different speeds and directions and allowing the user to verify them	
TC002	Absolute Focus Motion	Tests the absolute focus motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them	
TC003	Continuous Focus Motion with Timeout	Tests the continuous focus motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera	

CCTV Test Cases				
ID	Title Description			
TC004	Continuous Focus Motion with Stop	Tests the continuous focus motion of the camera by moving the camera and using the stop command to stop movement		
	Globa	al Configuration		
TC001	Retrieve Module Table	This procedure retrieves the module table, and allows the tester to verify that the DUT reports the proper type of device, manufacturer, model, and version.		
TC002	Global Set ID	This procedure ensures that a change to a static database object produces a change in globalSetIDParmeter.		
Iris				
TC001	Delta Iris Motion	Tests the delta iris motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera		
TC002	Absolute Iris Motion	Tests the absolute iris motion of the camera by moving the camera with two different speeds and directions and allowing the user to verify them		
TC003	Continuous Iris Motion with Timeout	Tests the continuous iris motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera		
TC004	Continuous Iris Motion with Stop	Tests the continuous iris motion of the camera by moving the camera and using the stop command to stop movement		
Label				
TC001	Get and Set Label	Verifies the number of labels the device can store. Test labels are stored in the device to verify storage capabilities		
TC002	Display Camera Location	Tests the capability of the device to display a text label on the video output		

CCTV Test Cases					
ID	Title Description				
		Lens			
TC001	Get Availability of Lens Equipment	Identifies and verifies the availability of equipment attached to the camera			
TC002	Control Auto Iris	Enables and disables the auto iris while the user verifies			
TC003	Control Auto Focus	Enables and disables the auto focus while the user verifies			
	Menu				
TC001	Menu	Tests the sending of menu commands to the CCTV while user verifies			
	Pan				
TC001	Delta Pan Motion	Tests the delta panning motion of the camera by moving the camera with two different speeds and direction and allowing the user to verify them			
TC002	Absolute Pan Motion	Tests the absolute panning motion of the camera by moving the camera with two different speeds and directions allowing the user to verify them			
TC003	Continuous Pan Motion with Timeout	Tests the continuous panning motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera			
TC004	Continuous Pan Motion with Stop	Tests the continuous panning motion of the camera by moving the camera and using the stop command to stop movement			
	Security				
TC001	Change Administrator Community Name	Verifies that the administrator can change the administrator community name stored in the DUT and properly affects operations			
TC002	Change User Community Name	Verifies that the administrator can change the user community names and their masks stored in the DUT and properly affects operations			

CCTV Test Cases					
ID	ID Title Description				
		Tilt			
TC001	Delta Tilt Motion	Tests the delta tilt motion of the camera by moving the camera with two different speeds and directions and allowing the user to verify them			
TC002	Absolute Tilt Motion	Tests the absolute tilt motion of the camera by moving the camera with two different speeds and directions and allowing the user to verify them			
TC003	Continuous Tilt Motion with Timeout	Tests the continuous tilt motion of the camera by moving the camera twice with the continuous command using the timeout parameter to stop the camera			
TC004	Continuous Tilt Motion with Stop	Tests the continuous tilting motion of the camera by moving the camera twice while using the stop command to stop movement			
		Zone			
TC001	Preset Position	Tests ability of the camera to store and move to preset camera positions			
TC002	Get-Set Zone	Tests ability of the camera to store camera zones			
TC003	Move In and Out of Zone	Tests the labeling capability of zones by moving to areas within zones			
		Zoom			
TC001	Delta Zoom Motion	Tests the delta zoom motion of the camera by moving the camera with two different speeds and directions and allowing the user to verify them			
TC002	Absolute Zoom Motion	Tests the absolute zoom motion of the camera by moving the camera with two different speeds and directions and allowing the user to verify them			
TC003	Continuous Zoom Motion with Timeout	Tests the continuous zoom motion of the camera by moving the camera twice with the continuous command using the timeout parameter to stop the camera			

CCTV Test Cases				
ID Title Description				
TC004	Continuous Zoom Motion with Stop	Tests the continuous zoom motion of the camera by moving the camera twice using the stop command		
		to stop movement		

TEST CASES

The details of each test case follow. The basic format of the test cases comes from the template that appears in NTCIP 8007-Test. As presented here the test case format has additional fields identifying constants and version history.

CCTV PRL Information

Test Case:	Title:	CCTV PRL Informa	tion	1	
PRL-TC001	Description:	This procedure retrie	eves	minimum project requirem	nents and
		maximum values, ch	neck	s for whether the required	objects are
		implemented, and p	erfoi	rms a sampling of the supp	orted values
		for P and C objects.			
	Constants:				
	Variables:				
	Pass/Fail	The DUT shall pass	eve	ry verification step included	d within the
	Criteria:	Test Case in order to	o pa	ss the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	CONFIGURE a list	<cctv and="" max="" sta<="" th=""><th>atic C</th><th>DIDs> that identifies</th><th></th></cctv>	atic C	DIDs> that identifies	
	read-only objects the	hat either define maxi	mur	n values that affect the	
	indexes of tables o	r static variables that	апе	ct limits on other	
0			r _ "	administrator"	
2.	EOR ObjectName	- ooch objectName in		CTV Max and Static	
Э.	OIDs>		1 ~ 0		
4.	GET ObjectNa	me			Pass/Fail
5.	Record the val	ue on the PRL			
6.	NEXT ObjectName)			
7.					
8.	CONFIGURE a list	of objectNames that	mus	st be supported	
9.	FOR [objectName	= each objectName ir	n Su	pported objectName List	
10.	FOR all possible [instance values]				
11.	GET [objectName].instance Pass/Fail				
		• • • •			
	Note: This loc	op performs the equiv	alen	t of a MIB Walk but uses	
10					
12.					
10. 1 <i>1</i>		;] _ <cct\ \="" alues<="" test="" th=""><th>> the</th><th>t identifies instances of</th><th></th></cct\>	> the	t identifies instances of	
14.	objects to test and	a value in which to te	st th	ne object with	
15.	FOR [objectName]	nstancel = each obie	ctNa	meInstance in <cctv< th=""><th></th></cctv<>	
	Test Values>				
16.	GET [objectNa	melnstance]			Pass/Fail
17.	RECORD RES	PONSE VALUE in [c	urre	ntValue]	
18.	FOR [testValue	e] = each objectName	Valu	ue in <cctv test<="" th=""><th></th></cctv>	
	Values>				
19.	SET [object	tNameInstance] = [te	stVa	alue]	Pass/Fail
20.	Record Ob	jectName, TestValue	, an	d errorStatus	
21.	NEXT TestValue				
22.	SET [objectNameInstance] = [currentValue] Pass/Fail				
23.	NEXT ObjectName				
24.	RECORD response	es on NTCIP PRL and	d no	te any anomalies	
		Test Case Result	ts		
Tested By:		Date	ad.		Pass/Fail
Test Case Notes:		18816	. u.		
Version History	V1 0 00/20/05 Init	ial Draft _ DDD			
	V1 1 02/27/06 Jm	nai Diail – RDR	nroc	ofed – .I.I	
		signification script allu	PIOC		

Cabinet Alarm

Test Case:	Title:	Cabinet Alar	m		
Alarm-TC001	Description: This Test Case tests the cabinet open alarm and label				
	associated with it.				
	Variables:				
	Pass/Fail	The DUT sha	Il pass every	verification step inclu	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	SET				Pass/Fail
	labelText. <alarn< th=""><th>nCLabIndex> =</th><th><alarmclab< th=""><th>Text1></th><th></th></alarmclab<></th></alarn<>	nCLabIndex> =	<alarmclab< th=""><th>Text1></th><th></th></alarmclab<>	Text1>	
	labelHeight. <ala< th=""><th>rmCLabIndex></th><th>= <alarmcla< th=""><th>abHeight1></th><th></th></alarmcla<></th></ala<>	rmCLabIndex>	= <alarmcla< th=""><th>abHeight1></th><th></th></alarmcla<>	abHeight1>	
	labelColor. <alar< th=""><th>mCLabIndex> :</th><th>= <alarmclal< th=""><th>bColor1></th><th></th></alarmclal<></th></alar<>	mCLabIndex> :	= <alarmclal< th=""><th>bColor1></th><th></th></alarmclal<>	bColor1>	
	labelStartRow.<	alarmCLabinde	ex> = < alarm(JLabStartRow1>	
	labelStartColum	n.< <i>alarmCLabl</i>	ndex> =		
•		Column1>		00 00 00 00 00	Dece/Feil
۷.		Idex.0 to <alari< th=""><th>mCLabindex></th><th>> 00 00 00 00 00 00</th><th>Pass/Fall</th></alari<>	mCLabindex>	> 00 00 00 00 00 00	Pass/Fall
3.	USER VERIFY th	at no labels are	e being show	ſ	Pass/Fail
4.	SET alarmLatchC	lear.0 to 0x00			Pass/Fail
5.	Turn on the alarm	and USER VE	RIFY the lab	el for the alarm is	Pass/Fail
	shown				
6.	GET alarmStatus	.0 and alarmLa	tchStatus.0		Pass/Fail
7.	VERIFY RESPON	NSE VALUE			Pass/Fail
	alarmStatus = 0x80				
-	alarmLatchStatus = 0x80				
8.	SET alarmLatchClear.0 to 0x00				Pass/Fail
9.				Pass/Fail	
10.	VERIFT RESPONSE VALUE Pass/Fall				
	alarmStatus = 0x80				
11		IS = UX8U	alarm ia ahau	in and depativista	Deee/Feil
11.	the alarm		alarm is show	m and deactivate	Pass/Fall
12	USER VERIEY th	e lahel for the a	alarm is off		Pass/Fail
13	GET alarmStatus	0 and alarml a	tchStatus 0		Pass/Fail
10.	VERIEY Respons	e Value			Pass/Fail
	alarmStatus = 0	x00			1 400/1 41
	alarmLatchStatu	is = 0x80			
15.	USER VERIEV the label for the alarm is off Pass/Fail			Pass/Fail	
16.	SET alarmLatchClear.0 to 0x00			Pass/Fail	
17.	GET alarmStatus.0 and alarmLatchStatus.0			Pass/Fail	
18.	VERIFY RESPOR	NSE VALUE			Pass/Fail
	alarmStatus = 0x00				
	alarmLatchStatu	is = 0x00			
		Test Case	Results		
Tested By:			Date		Pass/Fail
.			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 Initial Draft – RDR				
	V1.1 11/03/05 Removed deprecated labelFontType – RDR				
	VI.Z UZ/Z//U6 Ir	npiementea sci	ipi and proof	eu – JJ	

Enclosure Alarm

Test Case:	Title:	Enclosure Al	arm		
Alarm-TC002	Description: This Test Case tests the enclosure alarm and label associated				
		with it.			
	Variables:				
	Pass/Fail	The DUT sha	II pass every	verification step inclu	uded within the
	Criteria:	Test Case in o	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	SET				Pass/Fail
	labelText. <alarn< td=""><td>nELabIndex> =</td><td><alarmelab< td=""><td>Text1></td><td></td></alarmelab<></td></alarn<>	nELabIndex> =	<alarmelab< td=""><td>Text1></td><td></td></alarmelab<>	Text1>	
	labelHeight. <ala< td=""><td>rmELabIndex></td><td>= <alarmela< td=""><td>abHeight1></td><td></td></alarmela<></td></ala<>	rmELabIndex>	= <alarmela< td=""><td>abHeight1></td><td></td></alarmela<>	abHeight1>	
	labelColor. <alar< td=""><td>mELabIndex> =</td><td>= <alarmelat< td=""><td>oColor1></td><td></td></alarmelat<></td></alar<>	mELabIndex> =	= <alarmelat< td=""><td>oColor1></td><td></td></alarmelat<>	oColor1>	
	labelStartRow.<	alarmELabInde	x > = < a larm E	=LabStartRow1>	
	labelStartColum	n.< <i>alarmELabli</i>	ndex> =		
•		Column1>	ارم		Deee/Feil
2.		10ex.0 to 00 <a< td=""><td>larmELabind</td><td>ex> 00 00 00 00</td><td>Pass/Fall</td></a<>	larmELabind	ex> 00 00 00 00	Pass/Fall
		at na labala are	haina ahau		Dece/Feil
ى. 	USER VERIFT (I)		e being showi		Pass/Fall
<u> </u>	SET alamiLatoric		DIEV the Joh	al far tha alarm ia	Pass/Fall
Э.	shown	I ANU USER VE			Pass/Fall
6	GET alarmStatus	0 and alarml a	tchStatue 0		Dass/Fail
7		.0 and alamica	เป้าอเลเนร.บ		Pass/Fail
1.	alarmStatus = 0				r ass/r all
	alarmi atchStatu	x = 0x40			
8	SET alarmi atchClear 0 to 0x00			Pass/Fail	
9	GET alarmStatus 0 and alarmI atchStatus 0			Pass/Fail	
10.	VERIFY RESPON		toniotataolo		Pass/Fail
	alarmStatus = 0	x40			
	alarmLatchStatu	is = 0x40			
11.	USER VERIFY th	e label for the a	alarm is show	n and deactivate	Pass/Fail
	the alarm				
12.	USER VERIFY the label for the alarm is off			Pass/Fail	
13.	GET alarmStatus.0 and alarmLatchStatus.0			Pass/Fail	
14.	VERIFY RESPON	NSE VALUE			Pass/Fail
	alarmStatus = 0	x00			
	alarmLatchStatu	is = 0x40			
15.	USER VERIFY the label for the alarm is off			Pass/Fail	
16.	SET alarmLatchClear.0 to 0x00			Pass/Fail	
17.	GET alarmStatus.0 and alarmLatchStatus.0			Pass/Fail	
18.	VERIFY RESPON	NSE VALUE			Pass/Fail
	alarmStatus = 0	x00			
	alarmLatchStatu	s = 0x00	-		
	1	Test Case	Results		
Tested By:			Date Tested:		Pass/Fail
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R		
-	V1.1 11/03/05 R	emoved depred	cated labelFo	ntType – RDR	
	V1.2 02/27/06 Ir	nplemented scr	ipt and proof	ed – JJ	

Video Loss Alarm

Test Case:	Title:	Video Loss A	Alarm		
Alarm-TC003	Description:	This Test Cas	se tests the vi	deo loss alarm and l	abel associated
		with it.			
	Variables:				
	Pass/Fail	The DUT sha	ll pass every	verification step inclu	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	SET				Pass/Fail
	labelText. <alarn< td=""><td>nVLabIndex> =</td><td><alarmvlab< td=""><td>Text1></td><td></td></alarmvlab<></td></alarn<>	nVLabIndex> =	<alarmvlab< td=""><td>Text1></td><td></td></alarmvlab<>	Text1>	
	labelHeight. <ala< td=""><td>rmVLabIndex></td><td>= <alarmvla< td=""><td>abHeight1></td><td></td></alarmvla<></td></ala<>	rmVLabIndex>	= <alarmvla< td=""><td>abHeight1></td><td></td></alarmvla<>	abHeight1>	
	labelColor. <alar< td=""><td>mvLabindex> =</td><td>= <alarmvlat< td=""><td>0001011></td><td></td></alarmvlat<></td></alar<>	mvLabindex> =	= <alarmvlat< td=""><td>0001011></td><td></td></alarmvlat<>	0001011>	
	labelStartRow. <alarmvlabindex> = <alarmvlabstartrow1></alarmvlabstartrow1></alarmvlabindex>				
	<pre>abelStartColum <pre>calarm\/l abStart</pre></pre>	11. Salari i VLavi Column 1 S	nuez> -		
2	SET alarmi abelir	dex 0 to 00 00	<pre>>>></pre>	Index> 00 00 00	Pass/Fail
۷.	00				r ass/r all
3.	USER VERIFY th	at no labels are	e being show	n	Pass/Fail
4.	SET alarmLatchC	lear.0 to 0x00			Pass/Fail
5.	Turn on the alarm	and USER VE	ERIFY the lab	el for the alarm is	Pass/Fail
	shown				
6.	GET alarmStatus	.0 and alarmLa	tchStatus.0		Pass/Fail
7.	VERIFY RESPON	NSE VALUE			Pass/Fail
	alarmStatus = 0	x20			
0	alarmLatchStatus = 0x20			Deee/Feil	
0. 0	SET alarmLatchClear.U to UXUU			Pass/Fail	
9. 10				Pass/Fail	
10.	alarmStatus = 0x20				
	alarml atchStatu	x = 0x20			
11	USER VERIEV the label for the alarm is shown and deactivate Pass/Fail			Pass/Fail	
	the alarm			i door dii	
12.	USER VERIFY the label for the alarm is off			Pass/Fail	
13.	GET alarmStatus	.0 and alarmLa	tchStatus.0		Pass/Fail
14.	VERIFY RESPONSE VALUE Pass/Fail			Pass/Fail	
	alarmStatus = 0x00				
	alarmLatchStatu	s = 0x20			
15.	USER VERIFY the label for the alarm is off			Pass/Fail	
16.	SET alarmLatchClear.0 to 0x00			Pass/Fail	
17.	GET alarmStatus.0 and alarmLatchStatus.0			Pass/Fail	
18.				Pass/Fall	
	alarmi atchStatu				
	alarmitatoriotato		Poculto		
Tostod By:		1631 0436	Data		Dass/Fail
Tested by.			Tested:		r ass/r an
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R		
-	V1.1 11/03/05 R	emoved depre	cated labelFo	ntType – RDR	
	V1.2 02/27/06 In	nplemented sci	ript and proof	ed – JJ	

Temperature Alarm

Test Case:	Title:	Temperature	Alarm		
Alarm-TC004	Description: This Test Case tests the temperature alarm and label,				
		thresholds, ar	nd current val	ue associated with it.	
	Variables:				
	Pass/Fail	The DUT sha	ll pass every	verification step inclu	ided within the
-	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	SET	T I I I I I		- (4)	Pass/Fail
	label l ext. < alarn	11 Labindex> =		I ext1>	
		mTLobindex>	= <alarm112a< th=""><th>DHeight 1></th><th></th></alarm112a<>	DHeight 1>	
	labelStartRow <	alarmTl ablnde	- <alarm1 lai<="" th=""><th>TLabStartRow1></th><th></th></alarm1>	TLabStartRow1>	
	labelStartColumn < alarmTl ablindey> =				
	<alarmtlabstart< th=""><th>Column1></th><th>ind on</th><th></th><th></th></alarmtlabstart<>	Column1>	ind on		
2.	SET alarmLabellr	ndex.0 to 00 00	00 <alarmtl< th=""><th>_abIndex> 00 00 00</th><th>Pass/Fail</th></alarmtl<>	_abIndex> 00 00 00	Pass/Fail
3.	SET alarmTempe	ratureHighLow	Threshold.0 1	0	Pass/Fail
	<temperaturethree< th=""><th>eshold></th><th></th><th></th><th></th></temperaturethree<>	eshold>			
4.	USER VERIFY th	at no labels are	e being show	n	Pass/Fail
5.	SET alarmLatchC	lear.0 to 0x00			Pass/Fail
6.	Turn on the alarm	and USER VE	RIFY the lab	el for the alarm is	Pass/Fail
	shown				
7.	GET alarmStatus	.0 and alarmLa	tchStatus.0		Pass/Fail
8.	VERIFY RESPONSE VALUE Pass/Fail			Pass/Fail	
	alarmStatus = 0	x10			
0		S = 0X10			Doog/Eail
<u> </u>	GET alarmStatus 0 and alarmI atchStatus 0			Pass/Fail	
10.	VERIFY RESPONSE VALUE			Pass/Fail	
	alarmStatus = 0x10			1 455/1 41	
	alarmLatchStatu	is = 0x10			
12.	USER VERIFY th	e label for the a	alarm is show	n and deactivate	Pass/Fail
	the alarm				
13.	USER VERIFY the label for the alarm is off			Pass/Fail	
14.	GET alarmStatus.0 and alarmLatchStatus.0			Pass/Fail	
15.	VERIFY RESPON	NSE VALUE			Pass/Fail
	alarmStatus = 0	x00			
4.0	alarmLatchStatu	s = 0x10			
16.	USER VERIFY the label for the alarm is off			Pass/Fail	
17.	SET alarmLatchClear.0 to 0x00			Pass/Fall	
18.	GET alarmStatus.0 and alarmLatchStatus.0			Pass/Fall	
19.	alarmStatus - 0				Pass/Fail
	alarmi atchStatu	x = 0x00			
Tested By:		1001 0400	Date		Pass/Fail
rootoù by:			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 lr	nitial Draft – RD	K		
	V1.1 11/03/05 Re	eniovea deprec		nu ype – KDK	
	VI.Z UZ/Z//UO IN	npiementeu sci	ipi and proof	eu – JJ	

Pressure Alarm

Test Case:	Title:	Pressure Alarm	
Alarm-TC005	Description:	This Test Case tests the pressure alarm and la	bel, thresholds,
		and current value associated with it.	
	Variables:		
	Pass/Fail	The DUT shall pass every verification step inclu	uded within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
1.	SET		Pass/Fail
	labelText. <alarn< th=""><th>nPLabIndex> = <alarmplabtext1></alarmplabtext1></th><th></th></alarn<>	nPLabIndex> = <alarmplabtext1></alarmplabtext1>	
	labelHeight. <ala< th=""><th>rmPLabIndex> = <alarmplabheight1></alarmplabheight1></th><th></th></ala<>	rmPLabIndex> = <alarmplabheight1></alarmplabheight1>	
	labelColor. <alar< th=""><th>mPLabIndex> = <alarmplabcolor1></alarmplabcolor1></th><th></th></alar<>	mPLabIndex> = <alarmplabcolor1></alarmplabcolor1>	
	labelStartRow.<	alarmPLabIndex> = <alarmplabstartrow1></alarmplabstartrow1>	
	labelStartColum	n. <alarmplabindex> =</alarmplabindex>	
0		Column 1>	
Ζ.		Idex.0 10 00 00 00 00 <alarm@labindex> 00</alarm@labindex>	Pass/Fail
2	SET alarmProces	uroHighl ow/Throshold 0 to	Dass/Eail
5.	SET alamiritessu		F 855/F 811
_	USER VERIEY th	at no labels are being shown	Pass/Fail
	SET alarml atchC	Clear 0 to 0x00	Pass/Fail
6	Turn on the alarm	and USER VERIEY the label for the alarm is	Pass/Fail
0.	shown		1 400/1 41
7.	GET alarmStatus	.0 and alarmLatchStatus.0	Pass/Fail
8.	VERIFY RESPON	NSE VALUE	Pass/Fail
	alarmStatus = 0	x08	
	alarmLatchStatu	is = 0x08	
9.	SET alarmLatchC	Clear.0 to 0x00	Pass/Fail
10.	GET alarmStatus	.0 and alarmLatchStatus.0	Pass/Fail
11.	VERIFY RESPOR	NSE VALUE	Pass/Fail
	alarmStatus = 0	x08	
	alarmLatchStatu	is = 0x08	
12.	USER VERIFY th	e label for the alarm is shown and deactivate	Pass/Fail
	the alarm		
13.	USER VERIFY th	Pass/Fail	
14.	GET alarmStatus.0 and alarmLatchStatus.0		Pass/Fail
15.	VERIFY RESPO	NSE VALUE	Pass/Fail
4.0	alarmStatus = 0	x00 and alarmLatchStatus = $0x08$	
16.	VERIFY the label for the alarm is off		Pass/Fail
17.		Dear.0 to 0x00	Pass/Fall
18.			Pass/Fall
19.			Pass/Fall
	alarmi atchStatu		
	alarmilatoristatu		
Tootod Du:			Doog/Egil
Tested by.		Dale Tostod:	Fass/Faii
Tost Case Mater		1 53150.	
Test Case Notes.			
Version History	1/1 0 00/20/05 Ir	nitial Draft _ RDR	
v di sion i listory.	V1 1 11/03/05 D	emoved deprecated labelFontType _ RDP	
	V1 2 07/27/06 Ir	nplemented script and proofed	

Local Remote Alarm

Test Case:	Title:	Local Remote Alarm		
Alarm-TC006	Description:	Description: This Test Case tests the local-remote alarm and label		
		associated with it.		
	Variables:			
	Pass/Fail	The DUT shall pass every verification step inclu	ided within the	
	Criteria:	Test Case in order to pass the Test Case.		
Test Step	Test Procedure		Results	
Number				
1.	SET		Pass/Fail	
	labelText. <alarn< td=""><td>nLLabIndex> = <alarmllabtext1></alarmllabtext1></td><td></td></alarn<>	nLLabIndex> = <alarmllabtext1></alarmllabtext1>		
	labelHeight. <ala< td=""><td>rmLLabIndex> = <alarmllabheight1></alarmllabheight1></td><td></td></ala<>	rmLLabIndex> = <alarmllabheight1></alarmllabheight1>		
	labelColor. <alar< td=""><td>mLLabIndex> = <alarmllabcolor1></alarmllabcolor1></td><td></td></alar<>	mLLabIndex> = <alarmllabcolor1></alarmllabcolor1>		
	labelStartRow.<	alarmLLabIndex> = <alarmllabstartrow1></alarmllabstartrow1>		
	labelStartColum	n. <alarmllabindex> =</alarmllabindex>		
0		Joiumn1>		
<u> </u>		Idex.0 to 00 00 00 00 00 <alarminizabilidex> 00</alarminizabilidex>	Pass/Fall	
<u> </u>	USER VERIFY (I)		Pass/Fall	
4. 	SET alarmitation	and USED VEDIEV the lebel for the elermin	Pass/Fail	
J.	shown	I and USER VERIFY the laber for the alarm is	Pass/Fail	
6	GET alarmStatus	0 and alarmI atchStatus 0	Pass/Fail	
7			Pass/Fail	
1.	alarmStatus = 0	x04	1 433/1 41	
	alarmLatchStatu	s = 0x04		
8.	SET alarmLatchC	Pass/Fail		
9.	GET alarmStatus	Pass/Fail		
10.	VERIFY RESPON	NSE VALUE	Pass/Fail	
-	alarmStatus = 0x04			
	alarmLatchStatu	is = 0x04		
11.	USER VERIFY th	e label for the alarm is shown and deactivate	Pass/Fail	
	the alarm			
12.	USER VERIFY th	Pass/Fail		
13.	GET alarmStatus	.0 and alarmLatchStatus.0	Pass/Fail	
14.	VERIFY RESPONSE VALUE Pass/Fail			
	alarmStatus = 0	x00		
	alarmLatchStatu	is = 0x04		
15.	USER VERIFY the label for the alarm is off P			
16.	SET alarmLatchC	Pass/Fail		
17.	GET alarmStatus	Pass/Fail		
18.			Pass/Fail	
	alarmStatus = 0			
	alaminitationistatu			
Tested Dr.				
Tested By:		Date Tested:	Pass/Faii	
Test Case Notes:				
Version History:	V1.0 09/20/05 Ir	nitial Draft – RDR		
	V1.1 11/03/05 R	emoved deprecated labelFontType – RDR		
	v1.2 02/27/06 In	nplemented script and proofed – JJ		
Washer Fluid Alarm

Test Case:	Title:	Washer Fluid	d Alarm		
Alarm-TC007	Description:	This Test Cas	se tests the w	asher fluid alarm and	d label,
		thresholds, a	nd current val	ue associated with it	
	Variables:				
	Pass/Fail	The DUT sha	II pass every	verification step inclu	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	SET				Pass/Fail
	labelText. <alarn< td=""><td>nWFLabIndex></td><td>= <alarmwf< td=""><td>LabText1></td><td></td></alarmwf<></td></alarn<>	nWFLabIndex>	= <alarmwf< td=""><td>LabText1></td><td></td></alarmwf<>	LabText1>	
	labelHeight. <ala< td=""><td>rmWFLabInde.</td><td>x> = <alarmv< td=""><td>/FLabHeight1></td><td></td></alarmv<></td></ala<>	rmWFLabInde.	x> = <alarmv< td=""><td>/FLabHeight1></td><td></td></alarmv<>	/FLabHeight1>	
	labelColor. <alar< td=""><td>mWFLabIndex</td><td>> = <alarmwl< td=""><td>FLabColor1></td><td></td></alarmwl<></td></alar<>	mWFLabIndex	> = <alarmwl< td=""><td>FLabColor1></td><td></td></alarmwl<>	FLabColor1>	
	labelStartRow.<	alarmWFLabIn	dex> =		
	<alarmwflabsta< td=""><td>irtRow1></td><td></td><td></td><td></td></alarmwflabsta<>	irtRow1>			
	labelStartColum	n.< <i>alarmWFLa</i>	bindex> =		
~		ntColumn1>			
Ζ.			00 00 00 00 00		Pass/Fall
		ex>	Thrashald 0 t	•	Deee/Feil
Э.	SET alamitwashe	HIUIUTIGIILOW	Threshold.0 t	0	Pass/Fail
Δ		u- at no labele ar	a baina show	0	Dass/Fail
<u> </u>	SET alarmi atch	at no labels and	e being show		Pass/Fail
6	Turn on the alarm	and LISER VE	RIFY the lah	el for the alarm is	Pass/Fail
0.	shown				1 433/1 41
7.	GET alarmStatus 0 and alarmI atchStatus 0			Pass/Fail	
8.	VERIFY RESPO	NSE VALUE			Pass/Fail
	alarmStatus = 0x02				
	alarmLatchStatu	s = 0x02			
9.	SET alarmLatchC	lear.0 to 0x00			Pass/Fail
10.	GET alarmStatus	.0 and alarmLa	tchStatus.0		Pass/Fail
11.	VERIFY RESPOR	NSE VALUE			Pass/Fail
	alarmStatus = 0	x02			
	alarmLatchStatu	is = 0x02			
12.	USER VERIFY th	e label for the	alarm is show	n and deactivate	Pass/Fail
	the alarm				
13.	USER VERIFY th	e label for the	alarm is off		Pass/Fail
14.	GET alarmStatus	.0 and alarmLa	tchStatus.0		Pass/Fail
15.	VERIFY RESPO	NSE VALUE			Pass/Fail
	alarmStatus = 0	x00			
	alarmLatchStatu	s = 0x02	-1		
16.		e label for the	alarm is off		Pass/Fall
17.	SET alarmLatchClear.0 to 0x00			Pass/Fall	
18.	GET alarmStatus.0 and alarmLatchStatus.0			Pass/Fall	
19.	VERIFT RESPUT	NSE VALUE	otobStatua -	0,00	Pass/Fail
Tastad Da		Test Case	Results		
i estea By:			Date Tested:		Pass/Fail
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R		
	V1.1 11/03/05 R	emoved depre	cated labelFo	ntType – RDR	
	V1.2 02/27/06 Ir	nplemented sc	ript and proof	ed – JJ	

Identify Device

Test Case:	Title:	Identify Device			
Config-TC001	Description:	tion: This Test Case ensures that the DUT contains valid information			
_		for the module make, model, and version number as well as			
		other related inforr	mation.		
	Variables:				
	Pass/Fail	The DUT shall pas	ss every verification	step include	d within the
	Criteria:	Test Case in order	r to pass the Test C	ase.	
Test Step	Test Procedure				Results
Number					
1.	GET globalMaxM	odules.0 = [globalM	axModules]		Pass/Fail
2.	VERIFY RESPOR	ISE VALUE >= <ree< th=""><th>q_globalMaxModul</th><th>es></th><th>Pass/Fail</th></ree<>	q_globalMaxModul	es>	Pass/Fail
3.	FOR moduleInde	k = 1 TO [globalMax	(Modules]		
4.	GET				Pass/Fail
	moduleDevice	Node.moduleIndex,			
	moduleMake.n	noduleIndex,			
	moduleModel.	noduleIndex,			
	moduleVersior	.moduleIndex, and			
	moduleType.m	oduleIndex.			
5.	VERIFY module	DeviceNode.module	Index returns		Pass/Fail
-	1.3.6.1.4.1.1206	4.2.7			
6.	USER VERIFY F	RESPONSE VALUE			Pass/Fail
	moduleMake.m	oduleIndex = correc	t manufacturer nar	ne,	
	moduleModel.n	oduleIndex = corre	ct module number,		
	moduleversion	moduleIndex = corr	ect version, and		
	module l ype.mo	oduleIndex = correct	t type of the tested	device.	
	Nata Nata Na				
	Note: Make, Mod	iel, and version are	text descriptions.	nere	
~		a z Types: naroware	e and sollware.		
1.	NEXT moduleind				
Test Case Results					
Tested By:		Dat	te stod:		Pass/Fail
Toot Coop Notoo:		783			
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	itial Draft – RDR			
	V1.1 02/22/06 Ir	nplemented script a	nd proofed – JJ		

Identify Preset Position Range

Test Case: Config-TC002	<i>Title</i> : Description:	Identify Preset Position Range This Test Case ensures that the device indicates that it supports the required number of preset positions.		
	Variables: Pass/Fail Criteria:	The DUT shall pass every verification step inclu Test Case in order to pass the Test Case.	uded within the	
Test Step Number	Test Procedure		Results	
1.	GET rangeMaximumPreset.0		Pass/Fail	
2.	VERIFY RESPOR	NSE VALUE >= <req_rangemaxpreset></req_rangemaxpreset>	Pass/Fail	

Test Case Results						
Tested By:		Date Tested:		Pass/Fail		
Test Case Notes:						
Version History:	V1.0 09/20/05 Initial Draft – RD V1.1 02/22/06 Implemented sci	R ript and proofe	ed – JJ			

Identify Pan Limits

Test Case:	Title:	Identify Pan Limits	
Config-TC003	Description:	This Test Case identifies and verifies the left ar	nd right panning
		limits and the home position of the device.	
	Variables:		
	Pass/Fail	The DUT shall pass every verification step inclu	uded within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
1.	GET rangePanRi rangePanLeftLim	ghtLimit.0 = [rangePanRightLimit] and it.0 = [rangePanLeftLimit]	Pass/Fail
2.	VERIFY RESPOR	NSE VALUE rangePanRightLimit >= 2ightLimit>	Pass/Fail
3.	VERIFY RESPO	NSE VALUE rangePanLeftLimit <=	Pass/Fail
	<req_rangepanl< th=""><th>eftLimit></th><th></th></req_rangepanl<>	eftLimit>	
4.	SET positionPan.	.0 to Mode: 2 (Absolute), Speed:	Pass/Fail
	<absolutepanspe< th=""><th>eed>, Position: 0, which is hex value 02</th><th></th></absolutepanspe<>	eed>, Position: 0, which is hex value 02	
	<absolutepanspe< th=""><th>eed> 00 00</th><th></th></absolutepanspe<>	eed> 00 00	
5.	SET positionTilt.0) to Mode: 2 (Absolute), Speed:	Pass/Fail
	<absolutepanspe< th=""><th>eed>, Position: 0, which is hex value 02</th><th></th></absolutepanspe<>	eed>, Position: 0, which is hex value 02	
	<absolutepanspe< th=""><th>eed> 00 00</th><th></th></absolutepanspe<>	eed> 00 00	
6.	Make note curren		
1.	SET positionPan.	.0 to Mode: 2 (Absolute), Speed: -	Pass/Fail
	<absolutepanspe< th=""><th>bedy, Position: [rangePanLettLimit], which is</th><th></th></absolutepanspe<>	bedy, Position: [rangePanLettLimit], which is	
0		osolulePariSpeed> [rangePariLenLinnit]	Doog/Eail
0. 0	SET positionDan	0 to Mode: 2 (Absolute) Speed:	Pass/Fall Dass/Fail
9.	SET positionPari.	o to Mode. 2 (Absolute), Speed.	Fass/Fall
	N2 <absolutepan< th=""><th>Speeds 00 00</th><th></th></absolutepan<>	Speeds 00 00	
10	USER VERIEY c	amera moved back to its home position	Pass/Fail
11	SET positionPan	0 to Mode: 2 (Absolute) Speed:	Pass/Fail
	<absolutepanspe< th=""><th>eed>. Position: [rangePanRightLimit], which is</th><th></th></absolutepanspe<>	eed>. Position: [rangePanRightLimit], which is	
	hex value 02 <ab< th=""><th>solutePanSpeed> [rangePanRightLimit]</th><th></th></ab<>	solutePanSpeed> [rangePanRightLimit]	
12.	USER VERIFY ca	amera panned to its right limit	Pass/Fail
		Test Case Results	
Tested By:		Date	Pass/Fail
		Tested:	
Test Case Notes:			
Version History:	V1.0 09/20/05 Ir	nitial Draft – RDR	
-	V1.1 02/22/06 Ir	nplemented script and proofed – JJ	

True North Offset

Test Case:	Title:	True North C	Offset		
Config-TC004	Description:	This Test Case ensures that the user can configure the true			
		north setting	in the camera	l.	
	Variables:				
	Pass/Fail	The DUT sha	II pass every	verification step inclu	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	GET rangeTrueN	orthOffset.0 an	d record valu	e to	Pass/Fail
	[rangeTrueNorthC	Offset]			
2.	SET rangeTrueNorthOffset.0 to <valid_rangetruenorthoffset></valid_rangetruenorthoffset>				
3.	VERIFY RESPON	NSE ERROR =	noError		Pass/Fail
4.	SET rangeTrueN	orthOffset.0 to			
	<alternate_range< td=""><td>TrueNorthOffse</td><td>et></td><td></td><td></td></alternate_range<>	TrueNorthOffse	et>		
5.	VERIFY RESPON	NSE ERROR =	badValue		Pass/Fail
6.	SET rangeTrueN	orthOffset.0 to	[rangeTrueNo	orthOffset]	Pass/Fail
		Test Case	Results		
Tested By:			Date		Pass/Fail
			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R		
	V1.1 02/22/06 In	nplemented sc	ript and proof	ed – JJ	

Identify Tilt Limits

Test Case:	Title:	Identify Tilt I	_imits		
Config-TC0005	Description:	This Test Case ensures that the device indicates that it supports up and down tilting limits of the device.			
	Variables:	-	-		
	Pass/Fail	The DUT sha	II pass every	verification step incl	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	GET rangeTiltDov	GET rangeTiltDownLimit.0 and rangeTiltUpLimit.0			Pass/Fail
2.	VERIFY RESPORT	VERIFY RESPONSE VALUE			Pass/Fail
	rangeTiltUpLimit	rangeTiltUpLimit >= <req_rangetiltuplimit></req_rangetiltuplimit>			
3.	VERIFY RESPORT	NSE VALUE			Pass/Fail
	rangeTiltDownLin	nit <= < <i>req_rar</i>	ngeTiltDownL	imit>	
		Test Case	Results		
Tested By:			Date Tested:		Pass/Fail
Test Case Notes:			1		
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R		
	V1.1 02/22/06 Ir	nplemented sc	ript and proof	ed – JJ	

Identify Zoom Limits

Test Case: Config-TC006	<i>Title</i> : Description: Variables: Pass/Fail	Identify Zoor This Test Cas the required z The DUT sha	n Limits se ensures th zoom limit. Il pass every	at the device indicate	es that it supports
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step Number	Test Procedure				Results
1.	GET rangeZoomLimit.0.			Pass/Fail	
2.	VERIFY RESPOR	NSE VALUE >=	<pre><req_range< pre=""></req_range<></pre>	ZoomLimit>	Pass/Fail
		Test Case	Results		
Tested By:			Date Tested:		Pass/Fail
Test Case Notes:					
Version History:	V1.0 09/20/05 Initial Draft – RDR V1.1 02/22/06 Implemented script and proofed – JJ				

Identify Focus Limits

Test Case: Config-TC007	<i>Title</i> : Description:	Identify Focus Limits This Test Case ensures that the device indicates that it supports the required focus limit.			
	Variables: Pass/Fail Criteria:	The DUT sha Test Case in	II pass every order to pass	verification step incluster the Test Case.	uded within the
Test Step Number	Test Procedure				Results
1.	GET rangeFocusLimit.0				Pass/Fail
2.	VERIFY RESPO	NSE VALUE >=	= <req_range< td=""><td>FocusLimit></td><td>Pass/Fail</td></req_range<>	FocusLimit>	Pass/Fail
		Test Case	Results		
Tested By:			Date Tested:		Pass/Fail
Test Case Notes:					
Version History:	V1.0 09/20/05 Initial Draft – RDR V1.1 02/22/06 Implemented script and proofed – JJ				

Identify Iris Limit

Test Case: Config-TC008	<i>Title</i> : Description: Variables: Pass/Fail	Identify Iris I This test ensu required iris li The DUT sha	_imit ures that the mit. Il pass every	device indicates that	it supports the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step Number	Test Procedure				Results
1.	GET rangelrisLim	GET rangelrisLimit.0			Pass/Fail
2.	VERIFY RESPOR	NSE VALUE >=	<pre><req_range< pre=""></req_range<></pre>	IrisLimit>	Pass/Fail
		Test Case	Results		
Tested By:			Date Tested:		Pass/Fail
Test Case Notes:					
Version History:	V1.0 09/20/05 Initial Draft – RDR V1.1 02/22/06 Implemented script and proofed – JJ				

Identify Pan-Tilt Step Angle Minimum

Test Case:	Title:	Identify Pan-Tilt Step Angle Minimum			
Config-TC009	Description:	This Test Case ensures that the device indicates that it supports			
		the pan and tilt step angle minimum of the de	vice.		
	Variables:				
	Pass/Fail	The DUT shall pass every verification step inc	cluded within the		
	Criteria:	Test Case in order to pass the Test Case.			
Test Step	Test Procedure		Results		
Number					
1.	GET rangeMinim	umPanStepAngle.0 and	Pass/Fail		
	rangeMinimumTil	tStepAngle.0			
2.	VERIFY RESPOR	NSE VALUE	Pass/Fail		
	rangeMinimumPa	InStepAngle <=			
	<req_rangeminin< td=""><td>numPanStepAngle></td><td></td></req_rangeminin<>	numPanStepAngle>			
3.	VERIFY RESPOR	NSE VALUE	Pass/Fail		
	rangeMinimumTil	tStepAngle <=			
	<req_rangeminim< td=""><td>numTiltStepAngle></td><td></td></req_rangeminim<>	numTiltStepAngle>			
		Test Case Results			
Tested By:		Date	Pass/Fail		
		Tested:			
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RDR			
-	V1.1 02/22/06 Ir	nplemented script and proofed – JJ			

Identify Zone Functions

Test Case: Config-TC010	<i>Title</i> : Description:	Identify Zone This test ensured zone	Functions ures that the functions.	device indicates that	it supports the
	Variables: Pass/Fail Criteria:	The DUT sha Test Case in	II pass every order to pass	verification step inclus the Test Case.	uded within the
Test Step Number	Test Procedure				Results
1.	GET zoneCameraEquipped.0			Pass/Fail	
2.	VERIFY RESPON	NSE VALUE AN	ND		Pass/Fail
	<req_zonecamer< td=""><td>raEquipped> =</td><td><req_zoneca< td=""><td>ameraEquipped></td><td></td></req_zoneca<></td></req_zonecamer<>	raEquipped> =	<req_zoneca< td=""><td>ameraEquipped></td><td></td></req_zoneca<>	ameraEquipped>	
		Test Case	Results		
Tested By:			Date Tested		Pass/Fail
Test Case Notes:					
Version History:	V1.0 02/07/06 Initial Draft – RDR V1.1 02/22/06 Implemented script and proofed – JJ				

Monitor Discrete Input

Test Case:	<i>Title</i> :	Monitor Discrete Input	ute and label
Discrete-1C001	Description:	associated with it.	uts and label
	Variables:		
	Pass/Fail	The DUT shall pass every verification step inclu	ided within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step Number	Test Procedure		Results
1.	SET		Pass/Fail
	labelText. <input< td=""><td>LabelIndex1> = <inputlabtext1></inputlabtext1></td><td></td></input<>	LabelIndex1> = <inputlabtext1></inputlabtext1>	
	labelHeight. <inp< td=""><td>putLabelIndex1> = <inputlabheight1></inputlabheight1></td><td></td></inp<>	putLabelIndex1> = <inputlabheight1></inputlabheight1>	
	labelColor.	ItLabelIndex1> = <inputlabcolor1></inputlabcolor1>	
	labelStartColum	n <innutl abelindex1=""> = <iinpulabstattrow1></iinpulabstattrow1></innutl>	
	<inputlabstartco< td=""><td>blumn1></td><td></td></inputlabstartco<>	blumn1>	
2.	SET byte <input1< td=""><td>> of inputLabelIndex.0 to <inputlabelindex1></inputlabelindex1></td><td>Pass/Fail</td></input1<>	> of inputLabelIndex.0 to <inputlabelindex1></inputlabelindex1>	Pass/Fail
3.	Turn off inputs an	d USER VERIFY no labels for the	Pass/Fail
	corresponding inp	outs are shown	
4.	SET inputLatchC	lear.0 to 0x00	Pass/Fail
5.	Turn on the input shown	and USER VERIFY the label for the input is	Pass/Fail
6.	GET inputStatus.	0 and inputLatchStatus.0	Pass/Fail
7.	VERIFY RESPON	NSE VALUE	Pass/Fail
	inputStatus & 2 [^]	(< <i>input1</i> >-1) = inputStatus	
	inputLatchStatus	s & 2 [^] (< <i>input1</i> >-1) = inputLatchStatus	
	Noto:		
	This test will w	erify that hits < <i>input1</i> > are on	
8.	USER VERIFY th	le label for the corresponding input is on	Pass/Fail
9.	SET inputLatchC	lear.0 to 0x00	Pass/Fail
10.	GET inputStatus.	0 and inputLatchStatus.0	Pass/Fail

11.	VERIFY RESPONSE VALUE inputStatus & 2 ⁽ < <i>input1</i> >-1) = inputStatus	Pass/Fail			
	inputLatchStatus & 2 ⁽ < <i>input1</i> >-1) = inputLatchStatus				
	Note:				
	This test will verify that bits < <i>input</i> > are on.				
12.	USER VERIFY the label for the input is shown and deactivate the input	Pass/Fail			
13.	GET inputStatus.0 and inputLatchStatus.0	Pass/Fail			
14.	VERIFY RESPONSE VALUE	Pass/Fail			
	inputStatus & $2^{((input)-1)} = 0$				
	InputLatenStatus & 2 ⁽ (< <i>Input1></i> -1) = InputLatenStatus				
	Note:				
	This test will verify that bits < <i>input1</i> > are on (1) or off (0).				
15.	USER VERIFY the label for the input is off	Pass/Fail			
16.	SET inputLatchClear.0 to 0x00.	Pass/Fail			
17.	GET inputStatus.0 and inputLatchStatus.0	Pass/Fail			
18.	VERIFY RESPONSE VALUE	Pass/Fail			
	inputStatus & $2^{((input)-1)} = 0$				
	inputLatchStatus & $2^{((input 1 > -1))} = 0$				
	Noto:				
	This test will verify that bits < <i>input1</i> > are off				
19.	USER VERIFY the label for the input is off	Pass/Fail			
	Test Case Results				
Tested By:	Date	Pass/Fail			
	Tested:				
Test Case Notes:					
Version History:	V1.0 09/20/05 Initial Draft – RDR				
	V1.1 11/03/05 Removed deprecated labelFontType				
	Corrected comparison values for inputStatus and inputLato	hStatus – RDR			
	V1.2 U2/27/U6 Implemented script and proofed – JJ				

Monitor Discrete Output

Test Case:	Title:	Monitor Discrete Output	
Discrete-TC002	Description [.]	This Test Case verifies the state of discrete out	puts and label
		associated with it	
	Variables:		
	Pass/Fail	The DUT shall pass every verification step inclu	ided within the
	Criteria:	Test Case in order to pass the Test Case	
Test Sten	Test Procedure		Results
Number			nesuns
1	GET		Pass/Fail
	labelText <outou< th=""><th><i>itl abellnd</i>ex1> = [labelText1]</th><th>1 400/1 41</th></outou<>	<i>itl abellnd</i> ex1> = [labelText1]	1 400/1 41
	labelHeight <out< th=""><th>$t_{nutl} = t_{nutl} = t_{nutl} = t_{nutl}$</th><th></th></out<>	$t_{nutl} = t_{nutl} = t_{nutl} = t_{nutl}$	
	labelColor <outr< th=""><th>put abelindex = [labelColor1]</th><th></th></outr<>	put abelindex = [labelColor1]	
	labelStartRow <	output abellndex1> = [labelStartRow1]	
	labelStartColum	n < output l abellndex 1 > = [labelStartColumn1]	
2	GFT		Pass/Fail
<i>2</i> .	labelText <outou< th=""><th>itl.abellndex2> = [labelText2]</th><th>1 455/1 41</th></outou<>	itl.abellndex2> = [labelText2]	1 455/1 41
	labelHeight <out< th=""><th>t_{tnut} abellndex2> = [labelHeight2]</th><th></th></out<>	t_{tnut} abellndex2> = [labelHeight2]	
	labelColor <outr< th=""><th>putl abellndex2> = [labelColor2]</th><th></th></outr<>	putl abellndex2> = [labelColor2]	
	labelStartRow <	output abelindex2> = [labelStartRow2]	
	labelStartColum	n < output labellndex2> = [labelStartColumn2]	
3	SET		Pass/Fail
0.	labelText <outou< th=""><th>ıtl abellndex1> = <outputl abtext1=""></outputl></th><th>1 466/1 41</th></outou<>	ıtl abellndex1> = <outputl abtext1=""></outputl>	1 466/1 41
	labelHeight <out< th=""><th>tout abelindex $1 > = < 0.0000 \text{ abelindex} 1 > 0.0000 \text{ abelindex} 1 > 0.000000000000000000000000000000000$</th><th></th></out<>	tout abelindex $1 > = < 0.0000 \text{ abelindex} 1 > 0.0000 \text{ abelindex} 1 > 0.000000000000000000000000000000000$	
	labelColor <outr< th=""><th><math>put_abellndex1> = <output_abcolor1></output_abcolor1></math></th><th></th></outr<>	$put_abellndex1> = $	
	labelStartRow <	output abelindex 1> = < $output$ abStartRow1>	
	labelStartColum	n < output l abellndex 1> =	
	<output abstartc<="" th=""><th>Column1></th><th></th></output>	Column1>	
<u>4</u>	SFT		Pass/Fail
	labelText <outou< th=""><th>itl abellndex?> = <output! abtext?=""></output!></th><th>1 400/1 41</th></outou<>	itl abellndex?> = <output! abtext?=""></output!>	1 400/1 41
	labelHeight <out< th=""><th>toutl abellndex2> = <output! abheight2=""></output!></th><th></th></out<>	toutl abellndex2> = <output! abheight2=""></output!>	
	labelColor <outr< th=""><th>outl abelIndex2> = <output! abcolor2=""></output!></th><th></th></outr<>	outl abelIndex2> = <output! abcolor2=""></output!>	
	labelStartRow <	output abelindex2> = <output abstartrow2=""></output>	
	labelStartColum	n <output abellndex2=""> =</output>	
	<output abstartc<="" th=""><th>Column2></th><th></th></output>	Column2>	
5	SET outputContro	0.00000000000000000000000000000000000	Pass/Fail
6	SET byte <output< th=""><th>1^{-1} and $< output 2^{-1}$ of output abelindex 0 to</th><th>Pass/Fail</th></output<>	1^{-1} and $< output 2^{-1}$ of output abelindex 0 to	Pass/Fail
	<outputlabelinde< th=""><th>x1> and <outputlabelindex2></outputlabelindex2></th><th></th></outputlabelinde<>	x1> and <outputlabelindex2></outputlabelindex2>	
7.	SET outputContro	bl.0 to 0xXX 0x10	Pass/Fail
	Note:		
	The first byte	e, 0xXX, should be the value where only the bit	
	for <output1></output1>	is on. For example 0x04 for output1 = 3.	
8.	USER VERIFY th	at < <i>output1</i> > is on	Pass/Fail
9.	GET outputStatus	5.0	Pass/Fail
10.	VERIFY RESPON	ISE VALUE AND (2 [^] <output1>-1) =</output1>	Pass/Fail
	(2 ^{<} output1>-1)		
	, , ,		
	Note:		
	This test will ve	erify that bit < <i>output1</i> > is on.	
11.	SET outputContro	ol.0 to 0xXX 0x10	Pass/Fail
	Note:		
	The first byte,	0xXX, should be the value where only the bit	

	for < <i>output2</i> > is on			
12.	USER VERIFY that < output2> is	Pass/Fail		
13.	GET outputStatus.0			Pass/Fail
14.	VERIFY RESPONSE VALUE (2	<pre>^<output2>-1</output2></pre>) = (2^ <output2>-</output2>	Pass/Fail
	1).			
	Note:			
·	This test will verify that bit <c< td=""><td>output2> is on</td><td>•</td><td></td></c<>	output2> is on	•	
15.	SET outputControl.0 to 0x0000	~~		Pass/Fail
16.	USER VERIFY that all outputs a	ire off		Pass/Fail
17.	GET outputStatus.0			Pass/Fail
18.	VERIFY RESPONSE VALUE A	ND (2 [^] <outpu< td=""><td><i>it1</i>>-1) +</td><td>Pass/Fail</td></outpu<>	<i>it1</i> >-1) +	Pass/Fail
	$(2^{<}output_{2>-1}) = 0$			
19.	SEI JoholTovit covinuiti cholladovite		41	Pass/Fall
	labelHeight < outputLabelIndex 12		l] iabt11	
	labelColor < output abelIndex		ngnung pr11	
	labelStartRow < output abelling	/ev1> = [labelCold	StartRow1	
	labelStartColumn_ <outputlabel< td=""><td></td></outputlabel<>			
20.	SET			Pass/Fail
-	labelText. <outputlabelindex2></outputlabelindex2>	· = [labelText2	2]	
	labelHeight. <outputlabelindex< td=""><td>2> = [labelHe</td><td>ight2]</td><td></td></outputlabelindex<>	2> = [labelHe	ight2]	
	labelColor. <outputlabelindex2< td=""><td>> = [labelCold</td><td>or2]</td><td></td></outputlabelindex2<>	> = [labelCold	or2]	
	labelStartRow. <outputlabelinc< td=""><td>lex2> = [label</td><td>StartRow2]</td><td></td></outputlabelinc<>	lex2> = [label	StartRow2]	
	labelStartColumn. <outputlabe< td=""><td><i>llndex</i>2> = [la</td><td>belStartColumn2]</td><td></td></outputlabe<>	<i>llndex</i> 2> = [la	belStartColumn2]	
	Test Case	Results		
Tested By:		Date		Pass/Fail
		Tested:		
Test Case Notes:				
Version History:	V1.0.09/20/05 Initial Draft – PC)R		
v 6131011 1 113101 y.	VI.0 09/20/05 Illual Diall – KDK			
	V1.2 02/27/06 Implemented sc	ript and proof	ed – JJ	
	vi.2 02/27/06 implemented sc	ripi and proof	ea – JJ	

Get Availability of Equipment

Test Case:	Title:	Get Availabi	lity of Equip	nent	
Features-TC001	Description:	This Test Case identifies and verifies the availability of attached			
		equipment to	the camera.		
	Variables:				
	Pass/Fail	The DUT sha	Il pass every	verification step inclu	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	User Enter the av	ailable equipm	ent and store	it to	
	<req_systemcarr< td=""><td>neraEquipped></td><td></td><td></td><td></td></req_systemcarr<>	neraEquipped>			
2.	GET systemCam	eraEquipped.0			Pass/Fail
3.	VERIFY RESPON	NSE VALUE =	<req_system< td=""><td>CameraEquipped></td><td>Pass/Fail</td></req_system<>	CameraEquipped>	Pass/Fail
		Test Case	Results		
Tested By:			Date		Pass/Fail
			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R		
, 	V1.1 02/27/06 In	nplemented sc	ript and proof	ed – JJ	

Control Camera Power

Test Case:	Title:	Control Cam	era Power		
Features-TC002	Description:	This Test Cas	This Test Case enables and disables this feature while the user		
		verifies.			
	Variables:				
	Pass/Fail	The DUT sha	II pass every	verification step incl	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	GET systemCam	eraEquipped.0			Pass/Fail
2.	VERIFY that the	camera suppor	ts controlling	camera power	Pass/Fail
3.	GET systemCam	eraFeatureCon	trol.0		Pass/Fail
4.	SET systemCam	eraFeatureCon	trol.0 to 0x80	80	Pass/Fail
5.	DELAY 3 seconds				
6.	GET systemCameraFeatureStatus.0 Pass/Fail			Pass/Fail	
7.	VERIFY that bit 7 is on and camera power is ON Pass/Fa			Pass/Fail	
8.	SET systemCameraFeatureControl.0 to 0x8000 Pass/Fail			Pass/Fail	
9.	DELAY 3 second	S			
10.	GET systemCam	eraFeatureStat	us.0		Pass/Fail
11.	VERIFY that bit 7	' is off and cam	era power is	OFF	Pass/Fail
12.	SET systemCam	eraFeatureCon	trol.0 = 0x808	30	Pass/Fail
		Test Case	Results		
Tested By:			Date		Pass/Fail
-			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R		
	V1.1 11/03/05 Corrected systemCameraFeatureControl settings – RDR				
	V1.2 01/31/06 Added Delay to test – RDR				
	V1.3 02/27/06 Ir	nplemented sci	ript and proof	ed – JJ	

Control Heater Power

Test Case:	Title:	Control Heat	ter Power		
Features-TC003	Description:	This Test Case enables and disables this feature while the user			
		verifies.			
	Variables:				
	Pass/Fail	The DUT sha	II pass every	verification step inclu	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	GET systemCam	eraEquipped.0			Pass/Fail
2.	VERIFY that the o	camera suppor	ts controlling	heater power	Pass/Fail
3.	GET systemCame	GET systemCameraFeatureControl.0 = [systemFeatureControl] Pass/Fa			Pass/Fail
4.	SET systemCame	SET systemCameraFeatureControl.0 to 0x40800 Pass/Fail			Pass/Fail
5.	GET systemCam	eraFeatureStat	us.0		Pass/Fail
6.	VERIFY that bit 6	is on and heat	ter power is C	N	Pass/Fail
7.	SET systemCame	eraFeatureCon	trol.0 to 0x40	00	Pass/Fail
8.	GET systemCame	eraFeatureStat	tus.0		Pass/Fail
9.	VERIFY that bit 6	is off and heat	ter power is C)FF	Pass/Fail
10.	SET systemCame	eraFeatureCon	trol.0 = [syste	mFeatureControl]	Pass/Fail
		Test Case	Results		
Tested By:			Date		Pass/Fail
			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 In	nitial Draft – RD	R		
-	V1.1 11/03/05 C	orrected system	mCameraFea	tureControl settings	– RDR
	V1.2 02/27/06 In	nplemented sc	ript and proof	ed – JJ	

Control Wiper

Test Case:	Title:	Control Wiper		
Features-TC004	Description:	This Test Case enables and disables this feature while the user		
		verifies.		
	Variables:			
	Pass/Fail	The DUT shall pass every	verification step inclu	ided within the
	Criteria:	Test Case in order to pass	the Test Case.	
Test Step	Test Procedure			Results
Number				
1.	GET systemCam	eraEquipped.0		Pass/Fail
2.	VERIFY that the camera supports controlling wiper			Pass/Fail
3.	GET systemCameraFeatureControl.0 = [systemFeatureControl]			Pass/Fail
4.	SET systemCameraFeatureControl.0 to 0x2080 P			Pass/Fail
5.	GET systemCam	eraFeatureStatus.0		Pass/Fail
6.	VERIFY that bit 5	is on and wiper is ON		Pass/Fail
7.	SET systemCame	eraFeatureControl.0 to 0x200	00	Pass/Fail
8.	GET systemCam	GET systemCameraFeatureStatus.0		
9.	VERIFY that bit 5 is off and wiper is OFF			Pass/Fail
10.	SET systemCame	eraFeatureControl.0 = [syste	mFeatureControl]	Pass/Fail
		Test Case Results		
Tested By:		Date		Pass/Fail
-		Tested:		

Test Case Notes:	
Version History:	V1.0 09/20/05 Initial Draft – RDR V1.1 11/03/05 Corrected systemCameraFeatureControl settings – RDR V1.2 02/27/06 Implemented script and proofed – JJ

Control Washer

Test Case:	Title:	Control Wash	her		
Features-TC005	Description:	This Test Cas	e enables an	d disables this featu	re while the user
		verifies.			
	Variables:				
	Pass/Fail	The DUT shal	I pass every	verification step inclu	uded within the
	Criteria:	Test Case in c	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	GET systemCam	eraEquipped.0 :	= [systemCa	meraEquipped]	Pass/Fail
2.	VERIFY that the	camera support	s controlling	washer	Pass/Fail
3.	GET systemCam	eraFeatureCont	trol.0		Pass/Fail
4.	SET systemCameraFeatureControl.0 to 0x1080			Pass/Fail	
5.	GET systemCameraFeatureStatus.0			Pass/Fail	
6.	VERIFY that bit 4	is on and wash	ner is ON		Pass/Fail
7.	SET systemCame	SET systemCameraFeatureControl.0 to 0x1000			Pass/Fail
8.	GET systemCam	GET systemCameraFeatureStatus.0 Pass/Fail			Pass/Fail
9.	VERIFY that bit 4	is off and wash	ner is OFF		Pass/Fail
10.	SET systemCam	eraFeatureCont	rol.0 =		Pass/Fail
	[systemCameraE	quipped]			
		Test Case I	Results		
Tested By:			Date		Pass/Fail
			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RDI	R		
	V1.1 11/03/05 Corrected systemCameraFeatureControl settings – RDR				
	V1.2 02/27/06 Ir	mplemented scri	ipt and proof	ed – JJ	

Control Blower

Test Case:	Title:	Control Blower	
Features-TC006	Description:	This Test Case enables and disables this feature verifies.	re while the user
	Variables:		
	Pass/Fail	The DUT shall pass every verification step inclu	ided within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
1.	GET systemCam	eraEquipped.0	Pass/Fail
2.	VERIFY that the	camera supports controlling blower	Pass/Fail
3.	GET systemCam	eraFeatureControl.0 = [systemFeatureControl]	Pass/Fail
4.	SET systemCame	eraFeatureControl.0 to 0x0880	Pass/Fail
5.	GET systemCam	eraFeatureStatus.0	Pass/Fail
6.	VERIFY that bit 3	is on and blower is ON	Pass/Fail
7.	SET systemCame	eraFeatureControl.0 to 0x0800	Pass/Fail

8.	GET systemCameraFeatureStat	Pass/Fail			
9.	VERIFY that bit 3 is off and blow	ver is OFF	Pass/Fail		
10.	SET systemCameraFeatureCon	trol.0 = [systemFeatureControl]	Pass/Fail		
	Test Case Results				
Tested By:		Date Tested:	Pass/Fail		
Test Case Notes:					
Version History:	V1.0 09/20/05 Initial Draft – RDR V1.1 11/03/05 Corrected systemCameraFeatureControl settings – RDR V1.2 02/27/06 Implemented script and proofed – JJ				

Delta Focus Motion

Test Case:	Title:	Delta Focus	Motion				
Focus-TC001	Description:	This Test Cas	This Test Case tests the delta focus motion of the camera by				
	-	moving the ca	moving the camera with several different speed and direction				
		parameters a	nd allowing th	ne user to verify then	n.		
	Variables:						
	Pass/Fail	The DUT sha	Ill pass every	verification step inclu	uded within the		
	Criteria:	Test Case in	order to pass	the Test Case.			
Test Step	Test Procedure				Results		
Number							
1.	GET rangeFocus	sLimit.0			Pass/Fail		
2.	VERIFY camera	supports focus	limits		Pass/Fail		
3.	GET systemLens	FeatureContro	1.0 =		Pass/Fail		
	[systemLensFeat	tureControl]					
4.	SET systemLens	FeatureControl	.0 to 0x4000		Pass/Fail		
5.	SET positionFocusLens.0 to 01 < deltaFocusMoveSpeed>				Pass/Fail		
	<deltafocusmov< td=""><td>ement></td><td></td><td></td><td></td></deltafocusmov<>	ement>					
6.	USER VERIFY the camera lens moved towards far focus at the Pass/Fail						
	movement and s	peed specified	by the test va	riables			
	<pre><deltafocusmov< pre=""></deltafocusmov<></pre>	eSpeed> and <	delta-ocusM	ovement>			
7.	SET positionFoc	usLens.0 to 01	- <delta⊢ocus< td=""><td>MoveSpeed></td><td>Pass/Fail</td></delta⊢ocus<>	MoveSpeed>	Pass/Fail		
	<deltafocusmov< td=""><td>ement></td><td></td><td></td><td></td></deltafocusmov<>	ement>					
8.		ne camera lens	moved towar	ds near focus at	Pass/Fail		
	the movement an	nd speed specif	led by the tes	t variables			
		eSpeed> and <	deitaFocusiM	ovement>			
9.	SET systemLens		.0 =		Pass/Fall		
	IsystemLensFea		-				
	1	Test Case	Results	1			
Tested By:			Date		Pass/Fail		
			Tested:				
Test Case Notes:							
Version History:	V1.0 09/20/05 I	nitial Draft – RD)R				
	V1.1 02/06/06 A	Added step to tu	irn auto focus	on/off for test – JJ			
	V1.2 02/13/06 A	Added test for s	upport of focu	ıs limits – JJ			
	V1.3 02/27/06 I	6 Implemented script and proofed – JJ					

Absolute Focus Motion

Test Case:	Title:	Absolute Fo	cus Motion		
Focus-TC002	Description: This Test Case tests the absolute focus motion of the camera				
		by moving the	eed and direction		
		parameters a	nd allowing th	ne user to verify them	1.
	Variables:	P	J	, ,	
	Pass/Fail	The DUT sha	II pass everv	verification step inclu	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure		•		Results
Number					
1.	GET rangeFocus	Limit.0		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	Pass/Fail
2.	VERIFY camera	supports focus	limits		Pass/Fail
3.	SET positionFocu	usLens.0 to 02	00 00		Pass/Fail
4.	GET systemLens	FeatureControl	l = [systemLe	nsFeatureControl]	Pass/Fail
5.	SET systemLens	FeatureControl	.0 to 0x4000		Pass/Fail
6.	SET positionFocu	usLens.0 to 02	<absolutefoo< td=""><td>cusSpeed></td><td>Pass/Fail</td></absolutefoo<>	cusSpeed>	Pass/Fail
	<absolutefocusf< td=""><td>Position></td><td></td><td>-</td><td></td></absolutefocusf<>	Position>		-	
7.	USER VERIFY th	Pass/Fail			
	<absolutefocusposition></absolutefocusposition>				
8.	GET positionQue	Pass/Fail			
9.	VERIFY RESPONSE VALUE = <absolutefocusposition></absolutefocusposition>				Pass/Fail
10.	SET positionFocusLens.0 to 02 <absolutefocusspeed> Pass/F</absolutefocusspeed>				
	<absolutefocusf< td=""><td>Position2></td><td></td><td></td><td></td></absolutefocusf<>	Position2>			
11.	USER VERIFY the camera moved to the position defined by Pass/Fail				
12	GFT positionOue				Pass/Fail
13	VERIEY RESPON	NSE VALUE =	<absolutefor< td=""><td>Position2></td><td>Pass/Fail</td></absolutefor<>	Position2>	Pass/Fail
10.	SET systemLens	FeatureControl	0 =		Pass/Fail
17.	SciencensceatureControll Pass/Fall				
		Test Case	Results		
Tested By:		1001 0400	Date		Pass/Fail
rootou by:			Tested:		
Test Case Notes:			-		
			_		
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R		
	V1.1 11/03/05 Added test for positionQueryFocus – JJ				
	V1.2 02/06/06 Added step to turn auto focus on/off for test – RDR				
	VI.3 U2/13/06 A			IS IIIIIIIS	movemente
		p to set position			emovements
		mnlemented ec	rint and proo	fed – II	
	VI. 4 UZ/Z//UU I	inplemented SC	<u>and pioo</u>	160 - 00	

Continuous Focus Motion with Timeout

Test Case:	Title:	Continuous	Focus Motio	n with Timeout	
Focus-TC003	Description:	This Test Cas	se tests the co	ontinuous focus moti	on of the camera
		by moving the	e camera with	the continuous com	mand using the
		timeout parar	neter to stop	the camera.	
	Variables:				
	Pass/Fail	The DUT sha	II pass every	verification step inclu	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	GET systemLens	FeatureControl	l = [systemLe	nsFeatureControl]	Pass/Fail
2.	SET systemLens	FeatureControl	.0 to 0x4000		Pass/Fail
3.	GET timeoutFocu	us.0 = [timeoutF	Focus]		Pass/Fail
4.	SET timeoutFocus.0 to <alt_contfocustimeout></alt_contfocustimeout>			Pass/Fail	
5.	SET positionFocusLens.0 to 03 < conFocusSpeed > 00 00			Pass/Fail	
6.	USER VERIFY the camera lens stops moving in a far focus P				Pass/Fail
	direction after <alt_contfocustimeout> milliseconds</alt_contfocustimeout>				
7.	SET positionFocu	SET positionFocusLens.0 to 03 -< conFocusSpeed > 00 00			
8.	USER VERIFY th	ne camera lens	stops moving	in a near focus	Pass/Fail
	direction after <a< td=""><td>lt_contFocusTi</td><td><i>meout></i> millise</td><td>econds</td><td></td></a<>	lt_contFocusTi	<i>meout></i> millise	econds	
9.	SET timeoutFocu	is.0 back = [tim	eoutFocus]		Pass/Fail
10.	SET systemLens	FeatureControl	.0 =		Pass/Fail
	[systemLensFeat	ureControl]			
		Test Case	Results		
Tested By:			Date		Pass/Fail
			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R		
-	V1.1 02/06/06 A	dded step to tu	irn auto focus	on/off for test – JJ	
	V1.2 02/27/06 Ir	mplemented sci	ript and proof	ed – JJ	

Continuous Focus Motion with Stop

Test Case:	Title:	Continuous Focus Motion with Stop		
Focus-TC004	Description:	This Test Case tests the continuous focus motion of the camera by moving the camera and using the stop command to stop movement.		
	Variables:			
	Pass/Fail	The DUT shall pass every verification step inclu	uded within the	
	Criteria:	Test Case in order to pass the Test Case.		
Test Step	Test Procedure		Results	
Number				
1.	GET systemLens	FeatureControl = [systemLensFeatureControl]	Pass/Fail	
2.	SET systemLensl	FeatureControl.0 to 0x4000	Pass/Fail	
3.	GET timeoutFocu	is.0 = [timeoutFocus]	Pass/Fail	
4.	SET timeoutFocu	s.0 to 0	Pass/Fail	
5.	SET positionFocu	IsLens.0 to 03 < conFocusSpeed > 00 00	Pass/Fail	
6.	DELAY <alt_cont< td=""><td>FocusTimeout> milliseconds</td><td></td></alt_cont<>	FocusTimeout> milliseconds		
7.	SET positionFocusLens.0 to 00 00 00 00 Pass/Fail			
8.	USER VERIFY th	e camera stops moving	Pass/Fail	
9.	SET positionFocu	IsLens.0 to 03 - <confocusspeed> 00 00</confocusspeed>	Pass/Fail	

10.	DELAY <alt_contfocustimeout< th=""><th></th></alt_contfocustimeout<>					
11.	SET positionFocusLens.0 to 00	SET positionFocusLens.0 to 00 00 00 00				
12.	USER VERIFY the camera stop	s moving	Pass/Fail			
13.	SET timeoutFocus.0 back = [tim	eoutFocus]	Pass/Fail			
14.	SET systemLensFeatureControl	.0 =	Pass/Fail			
	[systemLensFeatureControl]	[systemLensFeatureControl]				
Test Case Results						
Tested By:	Date Pass/F					
	Tested:					
Test Case Notes:	Test Case Notes:					
Version History:	V1.0 09/20/05 Initial Draft – RDR					
	V1.1 02/06/06 Added step to turn auto focus on/off for test – JJ					
	V1.2 02/27/06 Implemented sc	ript and proofed – JJ				

Retrieve Module Table

Test Case:	Title:	Retrieve Module Table				
GloCon-TC001	Description [.]	This Test Case retrieves the module table and	allows the tester			
	Beeenpuon	to verify that the DUT reports the proper type of device.				
		manufacturer, model, and version.	,			
	Variables:					
	Pass/Fail	The DUT shall pass every verification step inclu	uded within the			
	Criteria:	Test Case in order to pass the Test Case.				
Test Step	Test Procedure	·	Results			
Number						
1.	GET globalMaxM	odules.0 = [globalMaxModules]	Pass/Fail			
2.	FOR N = 1 TO [g	lobalMaxModules]				
3.	GET the follow	/ing objects:	Pass/Fail			
	moduleNumb	per.N,				
	moduleDevic	eNode.N,				
	moduleMake	.N,				
	moduleMode	uleModel.N,				
	moduleVersio	/ersion.N, and				
	moduleType.	moduleType.N				
4.	VERIFY RESF	VERIFY RESPONSE VALUE for moduleNumber.N = N				
5.	VERIFY RESF	PONSE VALUE for moduleDeviceNode.N =	Pass/Fail			
	<oid devic<="" for="" th=""><th>e type></th><th></th></oid>	e type>				
	Note: Per NIC	CIP 8004 v01.37, the following are <oid for<="" th=""><th></th></oid>				
	device type> v					
	signal coi	troller = 1.3.6.1.4.1.1206.4.2.1				
	ramp con	troller = $1.3.6.1.4.1.1206.4.2.2$				
	ams	= 1.3.6.1.4.1.1206.4.2.3				
	tss	= 1.3.6.1.4.1.1206.4.2.4				
	ess	= 1.3.0.1.4.1.1200.4.2.5				
	CCIV	= 1.3.0.1.4.1.1200.4.2.7				
	dom	-1361411200.4.2.0				
	eem	= 1.3.6.1.4.1.1200.4.2.9				
	son	= 1.3.6.1.4.1.1200.4.2.10				
	network (amera = 1.3.6.1.4.1.1200.4.2.11				
	elms	= 1 3 6 1 4 1 1206 4 2 13				
	Cirio	1.0.0.1.1.1.1200.1.2.10				

6.	USER VERIFY RESPONSE indicates the manufacturer of <i>Note:</i> This might be the manufacturer of moduleType.N is hardware, of if the moduleType N is software	Pass/Fail		
7.	USER VERIFY RESPONSE indicates the correct model no	oduleModel.N component	Pass/Fail	
8.	USER VERIFY RESPONSE indicates the correct version r	VALUE for mo	oduleVersion.N e component	Pass/Fail
9.	USER VERIFY RESPONSE indicates the correct type <i>Note:</i> Values correspond to th other (1) hardware (2) software (3)	Pass/Fail		
10.	NEXT			
	Test Case	Results		
Tested By:		Date Tested:		Pass/Fail
Test Case Notes:				
Version History:	 V1.0 Original as defined in NTCIP 8007 v01.20 - B.3.2 Retrieve Module Table V2.0 02/22/06 Reformatted to use FOR and NEXT and simplified a number of steps. Changed "Change" to CONFIGURE Changed OID for device type to <oid device="" for="" type=""></oid> Removed "Temporary community name" from Variables field and adopted convention of using [name of variable] for the text description of local variables and using <name> for externally defined constants</name> Added list of moduleDeviceNodes from NTCIP 8004 – RDR V2.1 02/22/06 Implemented script and proofed – JJ 			

Global Set ID

Test Case:	Title:	Global Set ID			
GloCon-TC002	Description:	This procedure ensures that a change in a static database			
		object produces a change in globalSetIDParmeter.			
	Variables:				
	Pass/Fail	The DUT shall pass every verification step inclu	ided within the		
	Criteria:	Test Case in order to pass the Test Case.			
Test Step	Test Procedure		Results		
Number					
1.	GET globalSetID	Parmeter.0 = [globalSetIDParameter]	Pass/Fail		
2.	GET <setoctstrin< td=""><td>ngParameterOID> = [octStringOriginalValue]</td><td>Pass/Fail</td></setoctstrin<>	ngParameterOID> = [octStringOriginalValue]	Pass/Fail		
	<i>Note:</i> The < <i>setOctStringParameterOID</i> > can be any object with SYNTAX of OCTET STRING that is also is static database object.				
3.	SET < setOctStringParameterOID> = Pass/Fail < setOctStringParameterValue>				
4.	DELAY 120 Seco	onds			

ail ail ail ail ail ail					
ail ail ail ail ail ail					
ail ail ail ail ail ail					
ail ail ail ail ail ail ail					
ail ail ail ail ail					
ail ail ail ail					
ail ail ail					
ail ail					
ail					
Note: The exact frequency for updating the					
ail					
ail					
ail					
ail					

Delta Iris Motion

Test Case:	Title:	Delta Iris Motion			
Iris-TC001	Description [.]	This Test Case tests the delta iris motion of the	camera by		
	Dooonption	moving the camera with the delta command			
	Variables:				
	Pass/Fail	The DUT shall pass every verification step inclu	ided within the		
	Criteria:	Test Case in order to pass the Test Case			
Test Sten	Test Procedure		Results		
Number	1000110000000		noouno		
1	GET rangelrist im	it O	Pass/Fail		
2	VERIEV camera s	supports iris limits	Pass/Fail		
2.	CET avetemi and Easture Central = Javetemi and Easture Central Deco/Fail				
<u> </u>	GET systemiliens reactive Control – [systemiliens reactive Control] Pass/rail				
4.	SET systemLensFeatureControl.0 to 0x8000 Pass/Fail				
5.	SET positionIrisLe	ens.0 to 01 <deltairismovespeed></deltairismovespeed>	Pass/Fail		
	<deltalrismoveme< td=""><td>ent></td><td></td></deltalrismoveme<>	ent>			
6.	USER VERIFY th	e camera lens moved towards a closed	Pass/Fail		
	position at the mo	vement and speed specified by the test			
	variables <deltalr< td=""><td>isMoveSpeed> and <deltairismovement></deltairismovement></td><td></td></deltalr<>	isMoveSpeed> and <deltairismovement></deltairismovement>			
7.	SET positionIrisLe	ens.0 to 01 -< deltalrisMoveSpeed>	Pass/Fail		
	<deltalrismovement></deltalrismovement>				
8.	USER VERIEY the camera lens moved towards an open Pass/Fail				
	position at the mo	vement and speed specified by the test			
	variables <deltalr< td=""><td>isMoveSpeed> and <deltairismovement></deltairismovement></td><td></td></deltalr<>	isMoveSpeed> and <deltairismovement></deltairismovement>			
9	SET system ensi	FeatureControl 0 =	Pass/Fail		
0.	[evetem] eneFeat	ureControll	1 400/1 411		
	layatemicensi eat				

Test Case Results						
Tested By:			Date Tested:		Pass/Fail	
Test Case Notes:						
Version History:	V1.0 09/20/05 Initial Draft – RDR V1.1 02/06/06 Added step to turn auto iris on/off for test – RDR V1.2 02/13/06 Added test for support of iris limits – RDR V1.3 02/27/06 Implemented script and proofed – JJ					

Absolute Iris Motion

Test Case:	Title:	Absolute Iris	Motion			
Iris-TC002	Description: This Test Case tests the absolute iris motion of the camera by					
		moving the camera with several different speed				
		parameters a	nd allowing th	ne user to verify them	۱.	
	Variables:					
	Pass/Fail	The DUT sha	ll pass every	verification step inclu	uded within the	
	Criteria:	Test Case in	order to pass	the Test Case.		
Test Step	Test Procedure				Results	
Number						
1.	GET rangelrisLim	nit.O			Pass/Fail	
2.	VERIFY camera	supports iris lin	nits.		Pass/Fail	
3.	GET systemLens	FeatureContro	= [systemLe	nsFeatureControl]	Pass/Fail	
4.	SET systemLens	FeatureControl	.0 to 0x8000		Pass/Fail	
5.	SET positionIrisL	ens.0 to 02 <al< td=""><td>osoluteIrisSpe</td><td>ed></td><td>Pass/Fail</td></al<>	osoluteIrisSpe	ed>	Pass/Fail	
	<absoluteirisposi< td=""><td>ition></td><td></td><td></td><td></td></absoluteirisposi<>	ition>				
6.	USER VERIFY th	e camera mov	ed to the posi	tion defined by	Pass/Fail	
	<absoluteirisposi< td=""><td></td></absoluteirisposi<>					
7.	GET positionQue	rylris.0			Pass/Fail	
8.	VERIFY RESPON	Pass/Fail				
9.	SET positionIrisLe	ens.0 to 02 <al< td=""><td>osoluteIrisSpe</td><td>ed></td><td>Pass/Fail</td></al<>	osoluteIrisSpe	ed>	Pass/Fail	
	<absoluteirisposi< td=""><td></td></absoluteirisposi<>					
10.	USER VERIFY the camera moved to the position defined by				Pass/Fail	
	<absoluteirisposi< td=""><td>ition2></td><td></td><td></td><td></td></absoluteirisposi<>	ition2>				
11.	GET positionQue	Pass/Fail				
12.	VERIFY RESPON	NSE VALUE =	<absoluteirisl< td=""><td>Position2></td><td>Pass/Fail</td></absoluteirisl<>	Position2>	Pass/Fail	
13.	SET systemLens	Pass/Fail				
	[systemLensFeat	[systemLensFeatureControl]				
		Test Case	Results			
Tested By:			Date		Pass/Fail	
-			Tested:			
Test Case Notes:						
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R			
-	V1.1 11/03/05 Added tests for positionQueryIris – RDR					
	V1.2 02/06/06 A	dded step to tu	rn auto iris or	n/off for test – RDR		
	V1.3 02/13/06 A	dded test for s	upport of iris I	imits – RDR		
	V1.4 02/27/06 In	.4 02/27/06 Implemented script and proofed – JJ				

Continuous Iris Motion with Timout

Test Case:	Title:	Continuous	Iris Motion w	/ith Timeout		
Iris-TC003	Description:	This Test Cas	se tests the co	ontinuous iris motion	of the camera by	
		moving the ca	amera with th	e continuous comma	ind using the	
		timeout parar	neter to stop	the camera.		
	Variables:					
	Pass/Fail	The DUT sha	Ill pass every	verification step inclu	uded within the	
	Criteria:	Test Case in	order to pass	the Test Case.		
Test Step	Test Procedure				Results	
Number						
1.	GET systemLens	FeatureContro	I = [systemLe	nsFeatureControl]	Pass/Fail	
2.	SET systemLens	FeatureControl	.0 to 0x8000		Pass/Fail	
3.	GET timeoutIris.0) = [timeoutIris]			Pass/Fail	
4.	SET timeoutIris.0 to <alt_contiristimeout></alt_contiristimeout>			Pass/Fail		
5.	SET positionIrisLens.0 to 03 < conIrisSpeed > 00 00			Pass/Fail		
6.	USER VERIFY the camera lens stops moving towards a closed				Pass/Fail	
	position after <alt_contiristimeout> milliseconds</alt_contiristimeout>					
7.	SET positionIrisLens.0 to 03 - <conirisspeed> 00 00</conirisspeed>				Pass/Fail	
8.	USER VERIFY th	ne camera lens	stops moving	towards an open	Pass/Fail	
	position after <alt< td=""><td>t_contIrisTimeo</td><td>ut> millisecor</td><td>nds</td><td></td></alt<>	t_contIrisTimeo	ut> millisecor	nds		
9.	SET timeoutIris.0	= [timeoutIris]			Pass/Fail	
10.	SET systemLens	FeatureControl	.0 =		Pass/Fail	
	[systemLensFeat	ureControl]				
		Test Case	Results			
Tested By:			Date		Pass/Fail	
			Tested:			
Test Case Notes:						
Version History:		nitial Draft PC	D			
version i nistory.	V1 1 02/06/06 A	iniai Dian – NL Iddad stan to tu	un auto iris or	o/off for test - RDR		
	$\sqrt{1.1}$ $\sqrt{2}/\sqrt{00}$	nnlemented so	rint and proof	ad = 11		
	VI.Z UZ/ZI/UU II	inplemented SC	npt and proof	cu — JJ		

Continuous Iris Motion with Stop

Test Case:	Title:	Continuous	Iris Motion w	vith Stop		
Iris-TC004	Description:	This Test Cas	se tests the co	ontinuous iris motion	of the camera by	
		moving the ca	amera and us	ing the stop commar	nd to stop	
		movement.	movement.			
	Variables:					
	Pass/Fail	The DUT sha	II pass every	verification step inclu	ided within the	
	Criteria:	Test Case in	order to pass	the Test Case.		
Test Step	Test Procedure				Results	
Number						
1.	GET systemLens	FeatureControl	1.0 =		Pass/Fail	
	[systemLensFeat	ureControl]				
2.	SET systemLens	FeatureControl	.0 to 0x8000		Pass/Fail	
3.	GET timeoutIris.0	= [timeoutIris]			Pass/Fail	
4.	SET timeoutIris.0 to 0				Pass/Fail	
5.	SET positionIrisLens.0 to 03 < conIrisSpeed > 00 00				Pass/Fail	
6.	DELAY <alt_contiristimeout> milliseconds</alt_contiristimeout>					
7.	SET positionIrisLens.0 to 00 00 00 00				Pass/Fail	
8.	USER VERIFY the camera iris stops moving to a closed position				Pass/Fail	
9.	SET positionIrisLens.0 to 03 - <conirisspeed> 00 00</conirisspeed>				Pass/Fail	
10.	DELAY <alt_contlristimeout> milliseconds</alt_contlristimeout>					
11.	SET positionIrisLens.0 to 00 00 00 00				Pass/Fail	
12.	USER VERIFY th	e camera iris s	tops moving	to an open position	Pass/Fail	
13.	SET timeoutIris.0	= [timeoutIris]			Pass/Fail	
14.	SET systemLens	FeatureControl	.0 =		Pass/Fail	
	[systemLensFeat	ureControl]				
		Test Case	Results			
Tested By:			Date		Pass/Fail	
,			Tested:			
Test Case Notes:			•			
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R			
-	V1.1 02/20/06 A	dded step to tu	ırn auto iris or	n/off for test – RDR		
	V1.2 02/27/06 In	nplemented sc	ript and proof	ed – JJ		

Get and Set Label

Test Case:	Title:	Get and Set Label				
Label-TC0001	Description:	This Test Case verifies the number of labels the device can				
		store. Test labels are stored in the device to verify storage				
		capabilities.	ing etc.age			
	Variables:					
	Pass/Fail	The DLIT shall pass every verification step inclu	ided within the			
	Criteria:	Test Case in order to pass the Test Case				
Toot Stop	Toot Procedure		Populto			
Number	Test Procedure		Resuits			
	OFT and Chara la					
l. 0			Dece/Eeil			
<u> </u>		NSE VALUE >= <req_labeliviaximum></req_labeliviaximum>	Pass/Fall			
3.	FOR labelindex =					
4.	GET and Store	la deve - Dela IT - 441				
	label l ext.label	index = [iabei i ext],				
	labelHeight.lab	elindex = [labelHeight1],				
	labelColor.labe	lindex = [labelColor1],				
	labelStartRow.	abelindex = [labelStartRow1],				
	labelStartColur	nn.labelIndex = [labelStartColumn1]				
5.	SET		Pass/Fail			
	label l ext.label	ndex = <alt_label1ext>,</alt_label1ext>				
	labelHeight.lab	elIndex = <alt_labelheight>,</alt_labelheight>				
	labelColor.labe	IIndex = < <i>alt_labelColor</i> >,				
	labelStartRow.	labelIndex = <alt_labelstartrow>,</alt_labelstartrow>				
	labelStartColur	nn.labelIndex = <alt_labelstartcolumn></alt_labelstartcolumn>				
6.	GET					
	labelText.label	ndex,				
	labelHeight.lab	elIndex,				
	labelColor.labe	llndex,				
	labelStartRow.	labelIndex,				
	labelStartColur	nn.labelIndex				
7.	VERIFY RESPO	INSE VALUE	Pass/Fail			
	labelText = <al< th=""><th>t_labelText>,</th><th></th></al<>	t_labelText>,				
	labelHeight = <	alt_labelHeight>,				
	labelColor = <a< th=""><th>alt_labelColor>,</th><th></th></a<>	alt_labelColor>,				
	labelStartRow	= <alt_labelstartrow>,</alt_labelstartrow>				
	labelStartColur	nn = <alt_labelstartcolumn></alt_labelstartcolumn>				
8.	SET		Pass/Fail			
	labelText.label	ndex = [labelText1],				
	labelHeight.lab	elIndex = [labelHeight1],				
	labelColor.labe	IIndex = [labelColor1],				
	labelStartRow.	abelIndex = [labelStartRow1].				
	labelStartColur	nn.labelIndex = [labelStartColumn1]				
9.	NEXT labelIndex					
		Test Case Results				
Tested By:		Date	Pass/Fail			
100104 291		Tested:	r accir an			
Test Case Notes [.]						
Version History:	V1.0 09/20/05 Ir	iitial Draft – RDR				
. s. c. c	V1.1 11/03/05 R	emoved deprecated labelFontType from test – R	2DR			
	V1 2 11/08/05 P	emoved labelStatus Read Only object - RDP				
	V1.3 02/10/06 In	nnlemented scrint and proofed – 11				
1						

Display Camera Location

Test Case:	Title:	Display Camera Location			
Label-TC0002	Description:	This Test Case tests the capability to display a	text label on the		
	-	video output.			
	Variables:				
	Pass/Fail	The DUT shall pass every verification step incl	uded within the		
	Criteria:	Test Case in order to pass the Test Case.			
Test Step	Test Procedure		Results		
Number					
1.	GET and Store la	belMaximum.0 = [labelMaximum]			
2.	FOR labelIndex =	1 TO [labelMaximum]			
3.	GET and Store				
	labelText.label	Index = [labelText1],			
	labelHeight.lab	elIndex = [labelHeight1],			
	labelColor.labe	Index = [labelColor1],			
	labelStartRow.	labelIndex = [labelStartRow1],			
	labelStartColur	nn.labelIndex = [labelStartColumn1]			
4.	SET		Pass/Fail		
	labelText.label	Index = "Index: labelIndex",			
	labelHeight.lab	elIndex = <alt_loclabelheight>,</alt_loclabelheight>			
	labelColor.labe	olor.labelIndex = < <i>alt_locLabelColor</i> >,			
	labelStartRow.	labelIndex = <alt_loclabelstartrow>,</alt_loclabelstartrow>			
	labelStartColur	mn.labelIndex = <alt_loclabelstartcolumn></alt_loclabelstartcolumn>			
5.	SET labelLocati	onLabel.0 to labelIndex	Pass/Fail		
6.	SET labelEnable	eTextDisplay.0 to 0x80	Pass/Fail		
7.	USER VERIFY	hat the label is displayed	Pass/Fail		
8.	SET labelEnable	eTextDisplay.0 to 0x00	Pass/Fail		
9.	USER VERIFY	that the no labels are displayed	Pass/Fail		
10.	SET labelEnable	eTextDisplay.0 to 0x80	Pass/Fail		
11.	USER VERIFY	that the label is displayed	Pass/Fail		
	SET labelLocati	onLabel.0 to 0	Pass/Fail		
13.	USER VERIFY	that the label is not displayed	Pass/Fail		
14.	SET		Pass/Fail		
	labelText.label	Index = [labelText1],			
	labelHeight.lab	elIndex = [labelHeight1],			
	labelColor.labe	lIndex = [labelColor1],			
	labelStartRow.	labelIndex = [labelStartRow1],			
	labelStartColur	mn.labelIndex = [labelStartColumn1]			
15.	NEXT labelindex				
	1	Test Case Results			
Tested By:		Date	Pass/Fail		
		Tested:			
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RDR			
	V1.1 11/03/05 R	emoved deprecated labelFontType from test – F	RDR		
	V1.2 11/08/05 R	emoved labelStatus, Read Only			
	Altered la	belText to change for each index – RDR			
	V1.3 02/10/06 Ir	nplemented script and proofed – JJ			

Get Availability of Lens Equipment

Test Case: Lens-TC001	<i>Title</i> : Description:	Get Availability of Lens Equipment This Test Case identifies and verifies the availability of attached			
		equipment to the camera.			
	Variables:				
	Pass/Fail	The DUT sha	II pass every	verification step inclu	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step Number	Test Procedure				Results
1.	Enter the available equipment and store it to [reg_systemLensEquipped]				
2.	GET systemLens	Equipped.0			Pass/Fail
3.	VERIFY Respons	se equals [req_	systemLensE	quipped]	Pass/Fail
		Test Case	Results		
Tested By:			Date Tested:		Pass/Fail
Test Case Notes:					
Version History:	V1.0 09/20/05 Initial Draft – RDR				
	V1.1 02/27/06 Ir	nplemented sc	ript and proof	ed – JJ	

Control Auto Iris

Test Case:	Title:	Control Auto Iris					
Lens-TC002	Description:	This Test Cas	se enables an	nd disables this featu	re while the user		
		verifies.					
	Variables:						
	Pass/Fail	The DUT sha	Il pass every	verification step inclu	uded within the		
	Criteria:	Test Case in	order to pass	the Test Case.			
Test Step	Test Procedure				Results		
Number							
1.	GET systemLens	GET systemLensEquipped.0 = [systemLens]					
2.	VERIFY that the camera supports Auto Iris			Pass/Fail			
3.	GET systemLensFeatureControl.0			Pass/Fail			
4.	SET systemLensFeatureControl.0 to 0x8080			Pass/Fail			
5.	GET systemLensFeatureStatus.0			Pass/Fail			
6.	VERIFY that bit 7 is on and Auto Iris is ON			Pass/Fail			
7.	SET systemLensFeatureControl.0 to 0x8000			Pass/Fail			
8.	GET systemLens	FeatureStatus.	0		Pass/Fail		
9.	VERIFY that bit 7	is off and Auto	Iris is OFF		Pass/Fail		
10.	SET systemLens	FeatureControl	.0 = [systemL	ens]	Pass/Fail		
		Test Case	Results				
Tested By:			Date		Pass/Fail		
			Tested:				
Test Case Notes:							
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R				
, , , , , , , , , , , , , , , , , , ,	V1.1 11/03/05 C	Corrected system	mLensFeature	eControl – RDR			
	V1.2 02/27/06 Ir	mplemented sc	ript and proof	ed – JJ			

Control Auto Focus

Test Case:	Title:	Control Auto	Focus				
Lens-TC003	Description:	This Test Cas	se enables an	nd disables this featu	re while the user		
		verifies.					
	Variables:						
	Pass/Fail	The DUT sha	Il pass every	verification step inclu	uded within the		
	Criteria:	Test Case in	order to pass	the Test Case.			
Test Step	Test Procedure				Results		
Number							
1.	GET systemLens	Equipped.0 = [systemLens]		Pass/Fail		
2.	VERIFY that the o	VERIFY that the camera supports Auto Iris			Pass/Fail		
3.	GET systemLensFeatureControl.0			Pass/Fail			
4.	SET systemLensFeatureControl.0 to 0x4080			Pass/Fail			
5.	GET systemLensFeatureStatus.0			Pass/Fail			
6.	VERIFY that bit 7 is on and Auto Focus is ON			Pass/Fail			
7.	SET systemLens	ET systemLensFeatureControl.0 to 0x4000			Pass/Fail		
8.	GET systemLens	FeatureStatus.	0		Pass/Fail		
9.	VERIFY that bit 7	is off and Auto	Focus is OF	F	Pass/Fail		
10.	SET systemLens	FeatureControl	.0 = [systemL	ens]	Pass/Fail		
		Test Case	Results				
Tested By:			Date		Pass/Fail		
			Tested:				
Test Case Notes:							
Version History:	V1.0 09/20/05 Ir	V1.0 09/20/05 Initial Draft – RDR					
-	V1.1 11/03/05 C	orrected system	mLensFeature	eControl – RDR			
	V1.2 02/27/06 In	nplemented sci	ript and proof	ed – JJ			

Menu

Test Case:	Title: Menu					
Menu-TC001	Description: This Tes	st Case sends menu commands to the	CCTV.			
	Variables:					
	Pass/Fail The DU	T shall pass every verification step inc	luded within the			
	Criteria: Test Ca	se in order to pass the Test Case.				
Test Step	Test Procedure		Results			
Number						
1.	SET menuActivate.0 to 258	5	Pass/Fail			
2.	USER VERIFY that the me	nu is displayed	Pass/Fail			
3.	SET menuControl.0 to 1 (p	ageDown)	Pass/Fail			
4.	USER VERIFY that the me	nu moved down a page	Pass/Fail			
5.	SET menuControl.0 to 2 (p	ageUp)	Pass/Fail			
6.	USER VERIFY that the me	nu moved up a page	Pass/Fail			
7.	SET menuControl.0 to 4 (c	ursorDown)	Pass/Fail			
8.	USER VERIFY that the cur	sor moved down	Pass/Fail			
9.	SET menuControl.0 to 3 (c	ursorUp)	Pass/Fail			
10.	USER VERIFY that the cur	sor moved up	Pass/Fail			
11.	User Input: Is the cursor in	a position to move right?				
	If Yes GOTO Step 14					
	If No GOTO Step 12					
12.	SET menuControl.0 to 3,4	SET menuControl.0 to 3,4 or 9 depending on user input Pass/F				
13.	GOTO Step 11					
14.	SET menuControl.0 to 5 (c	Pass/Fail				
15.	USER VERIFY that the cur	Pass/Fail				
16.	SET menuControl.0 to 6 (c	Pass/Fail				
17.	USER VERIFY that the cur	Pass/Fail				
18.	User Input: Is the cursor in a position to enter a value?					
	If Yes GOTO Step 21					
	If No GOTO Step 19					
19.	SET menuControl.0 to 3,4,	5,6 or 9, depending on user input	Pass/Fail			
20.	GOTO Step 18					
21.	SET menuControl.0 to 7 (ir	ncrementValue)	Pass/Fail			
22.	USER VERIFY the value h	as been incremented	Pass/Fail			
23.	SET menuControl.0 to 8 (d	ecrementValue)	Pass/Fail			
24.	USER VERIFY the value h	as been decremented	Pass/Fail			
25.	SET menuControl.0 to 9 (e	nterValue)	Pass/Fail			
26.	USER VERIFY the value h	as been entered	Pass/Fail			
27.	SET menuActivate.0 to 0		Pass/Fail			
28.	USER VERIFY the menu h	as been deactivated	Pass/Fail			
29.	SET menuActivate.0 to <ac< th=""><th>ctivateMenuTimeout></th><th>Pass/Fail</th></ac<>	ctivateMenuTimeout>	Pass/Fail			
30.	USER VERIFY that the me	nu is displayed	Pass/Fail			
31.	Delay <activatemenutimeout> +1 second</activatemenutimeout>					
32.	USER VERIFY the menu h	as been deactivated	Pass/Fail			
	Test	Case Results				
Tested By:		Date	Pass/Fail			
		Tested:				
Test Case Notes:						

Version History:	V1.0 09/20/05 Initial Draft – RDR
	V1.1 11/28/05 – Changed options in step 11 to give user more choices – JJ
	V1.2 02/27/06 Implemented script and proofed – JJ

Delta Pan Motion

Test Case:	Title:	Delta Pan Mo	otion		
Pan-TC001	Description:	This Test Cas	se tests the de	elta panning motion of	of the camera by
		moving the ca	amera with se	veral different speed	and direction
		parameters a	nd allowing th	e user to verify them).
	Variables:	•	Ū		
	Pass/Fail	The DUT sha	Ill pass every	verification step inclu	ided within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	GET rangePanLe	eftLimit.0 and ra	angePanRight	Limit.0	Pass/Fail
2.	VERIFY camera	supports pannii	ng and pan lir	nits	Pass/Fail
3.	SET positionPan.	.0 to Mode: 1 ([Delta), Speed		Pass/Fail
	<deltapanmoves< td=""><td>peed>, Position</td><td>n: <deltapann< td=""><td><i>lovement></i>, which</td><td></td></deltapann<></td></deltapanmoves<>	peed>, Position	n: <deltapann< td=""><td><i>lovement></i>, which</td><td></td></deltapann<>	<i>lovement></i> , which	
	is hex value 01 <	deltaPanMoveS	Speed> <delta< td=""><td>aPanMovement></td><td></td></delta<>	aPanMovement>	
4.	USER VERIFY th	ne camera mov	ed in a clockw	ise direction at the	Pass/Fail
	movement and sp	beed specified	by <i><deltapan< i=""></deltapan<></i>	<i>Movement></i> and	
	<deltapanmoves< td=""><td>peed></td><td></td><td></td><td></td></deltapanmoves<>	peed>			
5.	SET positionPan.	.0 to Mode: 1 (I	Delta), Speed:	-	Pass/Fail
	<deltapanmoves< td=""><td>peed>, Position</td><td>n: <<i>deltaPan</i>M</td><td><i>lovement></i>, which</td><td></td></deltapanmoves<>	peed>, Position	n: < <i>deltaPan</i> M	<i>lovement></i> , which	
	is hex value 01 -<	<deltapanmove< td=""><td>Speed> <del< td=""><td>taPanMovement></td><td></td></del<></td></deltapanmove<>	Speed> <del< td=""><td>taPanMovement></td><td></td></del<>	taPanMovement>	
6.	USER VERIFY th	ne camera mov	ed in a counte	erclockwise	Pass/Fail
	direction at the m	ovement and s	peed specifie	d by	
	<deltapanmoverr< td=""><td>nent> and <delt< td=""><td>taPanMoveSp</td><td>eed></td><td></td></delt<></td></deltapanmoverr<>	nent> and <delt< td=""><td>taPanMoveSp</td><td>eed></td><td></td></delt<>	taPanMoveSp	eed>	
		Test Case	Results		
Tested By:			Date		Pass/Fail
			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD)R		
-	V1.1 11/11/05 A	dded instructio	ns to test scri	pt to help user execu	ite test – JJ
	V1.2 02/13/06 A	dded test for s	upport of pani	ning and pan limits –	JJ
	V1.3 02/22/06 Implemented script and proofed – JJ				

Absolute Pan Motion

Test Case	Title [.]	Absolute Pan	Motion		
Pan-TC002	Description:	This Test Case	e tests the al	osolute nannina moti	on of the camera
	Description.	by moving the	camera with	soveral different so	and direction
		parameters an	d allowing th	a user to verify them	
	Variables:	parameters an	iu allowing ti		
				varification atom inclu	ded within the
	Pass/Fall	The DUT shall	i pass every	the Test Case	ided within the
	Chiena:	Test Case in o	profer to pass	the rest case.	
Test Step	Test Procedure				Results
Number					
1.	GET rangePanLe	ftLimit.0 and rar	ngePanRight	Limit.0	Pass/Fail
2.	VERIFY camera s	supports pannin	g and pan lir	nits	Pass/Fail
3.	GET rangePanHo	mePosition.0 =	[rangePanH	omePostion]	Pass/Fail
4.	SET positionPan.	0 to Mode:2 (Ab	osolute), Spe	ed:	Pass/Fail
	<absolutepanspe< td=""><td>ed>, Position: [</td><td>rangePanHo</td><td>mePostion], which</td><td></td></absolutepanspe<>	ed>, Position: [rangePanHo	mePostion], which	
	is hex value 02 <a< td=""><td>absolutePanSpe</td><td>ed> [rangeF</td><td>PanHomePostion]</td><td></td></a<>	absolutePanSpe	ed> [rangeF	PanHomePostion]	
5.	Note that current	position as the I	Home positic	n	
6.	SET positionPan.	0 to Mode: 2(Ab	osolute), Spe	ed:	Pass/Fail
	<absolutepanspe< td=""><td>ed>, Position: <</td><td><absolutepai< td=""><td>nPosition>, which</td><td></td></absolutepai<></td></absolutepanspe<>	ed>, Position: <	<absolutepai< td=""><td>nPosition>, which</td><td></td></absolutepai<>	nPosition>, which	
	is hex value 02 <absolutepanspeed> <absolutepanposition>.</absolutepanposition></absolutepanspeed>				
7.	USER VERIFY the camera moved to the position Pass/Fail				
	<absolutepanpos< td=""><td></td></absolutepanpos<>				
8.	GET positionQue	Pass/Fail			
9.	VERIFY RESPONSE VALUE = <absolutepanposition></absolutepanposition>				Pass/Fail
10.	SET positionPan.0 to Mode: 2(Absolute), Speed:				Pass/Fail
	<absolutepanspeed>. Position: [rangePanHomePostion]. which</absolutepanspeed>				
	is hex value 02 <i><absolutepanspeed></absolutepanspeed></i> [rangePanHomePostion]				
11.	USER VERIEV the camera moved back to Home Position				Pass/Fail
12.	SET positionPan.	0 to Mode: 2(Ab	osolute). Spe	ed:	Pass/Fail
	<absolutepanspe< td=""><td>ed>. Position: <</td><td><absolutepar< td=""><td>Position2>, which</td><td></td></absolutepar<></td></absolutepanspe<>	ed>. Position: <	<absolutepar< td=""><td>Position2>, which</td><td></td></absolutepar<>	Position2>, which	
	is hex value $02 < 2$	absolutePanSne	ed> <absoli< td=""><td>itePanPosition2></td><td></td></absoli<>	itePanPosition2>	
13	USER VERIEY th	e camera move	d to the posi	tion	Pass/Fail
10.	<absolutepanpos< td=""><td>sition2></td><td></td><td></td><td>1 466/1 41</td></absolutepanpos<>	sition2>			1 466/1 41
14	GET positionQue	rvPan 0			Pass/Fail
15	VERIEY RESPON	VAIUF = <	ahsolutePar	Position2>	Pass/Fail
			Results		1 400/1 41
Tootod Py:		1031 04301	Data		Dace/Eail
Tested by.			Dale Tostod:		F 855/F 811
Taat Casa Nataa			Testeu.		
Test Case Notes:					
Varaian History:	1/1 0 00/20/05 In				
version mistory.		ddad taat far na	n NaitionOuoryE	Don II	
	V1.1 11/04/05 Added test for positionQueryPan – JJ				
	v I.Z I I/I I/US A	uueu sieps io se	er position to		
		ddad atan ta ma	wo hook to !	lomo offer first share	uto toot
	VI.3 UI/30/00 A	dded teet for an	DVE DACK LOF	ione alter IIIst absol	
	VI.4 U2/13/U6 A	uded test for su	pport of pani	ning and pan limits –	JJ
	v1.5 U2/22/U6 In	npiementea scri	ipi and proof	ea – JJ	

Continuous Pan Motion with Timeout

Test Case:	Title: Continuous Pan Motion with Timeout					
Pan-TC003	Description:	This Test Ca	se tests the c	ontinuous panning m	otion of the	
		camera by m	oving the can	nera with the continu	ous command	
		using the time	eout paramet	er to stop the camera	а.	
	Variables:	J				
	Pass/Fail	The DUT sha	The DUT shall pass every verification step included within the			
	Criteria:	Test Case in	order to pass	the Test Case.		
Test Step	Test Procedure				Results	
Number						
1.	GET rangePanLe	eftLimit.0 and ra	angePanRigh	tLimit.0	Pass/Fail	
2.	VERIFY camera	supports panni	ng and pan lii	nits	Pass/Fail	
3.	GET timeoutPan.	0 = [timeoutPa	n]		Pass/Fail	
4.	SET timeoutPan.	Pass/Fail				
5.	SET positionPan.0 to Mode: 3(Continuous), Speed:			Pass/Fail		
	<conpanspeed>, Position: 0, which is hex value</conpanspeed>					
	03 <conpanspeed> 00 00</conpanspeed>					
6.	USER VERIFY the camera stops moving in a clockwise Pass/Fail					
	direction after <alt_contpantimeout> milliseconds</alt_contpantimeout>					
7.	SET positionPan.	.0 to Mode: 3(C	Continuous), S	Speed: -	Pass/Fail	
	<conpanspeed>,</conpanspeed>	, Position: 0, wł	hich is hex va	lue		
	03 - <conpanspe< th=""><th>ed> 00 00</th><th></th><th></th><th></th></conpanspe<>	ed> 00 00				
8.	USER VERIFY th	ne camera stop	s moving in a	counterclockwise	Pass/Fail	
	direction after <a< td=""><td>lt_contPanTime</td><td><i>eout></i> millisec</td><td>onds</td><td></td></a<>	lt_contPanTime	<i>eout></i> millisec	onds		
9.	SET timeoutPan.	= [timeoutPan]			Pass/Fail	
		Test Case	Results			
Tested By:			Date		Pass/Fail	
			Tested:			
Test Case Notes:						
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD)R			
-	V1.1 11/11/05 A	dded instructio	ons to test scr	ipt to help user exect	ute test – JJ	
	V1.2 02/13/06 A	dded test for s	upport of pan	ning and pan limits –	- JJ	
	V1.3 02/22/06 Implemented script and proofed – JJ					

Continuous Pan Motion with Stop

Test Case:	Title:	Continuous Pan Motion	with Stop	
Pan-TC004	Description:	This Test Case tests the continuous panning motion of the		
	·	camera by moving the camera and using the stop command to		
		stop movement.	0	•
	Variables:			
	Pass/Fail	The DUT shall pass every	verification step inclu	uded within the
	Criteria:	Test Case in order to pass	the Test Case.	
Test Step	Test Procedure			Results
Number				
1.	GET rangePanLe	ftLimit.0 and rangePanRight	tLimit.0	Pass/Fail
2.	VERIFY camera	supports panning and pan lir	nits	Pass/Fail
3.	GET timeoutPan.0 = [timeoutPan]			Pass/Fail
4.	SET timeoutPan.0 to 0			Pass/Fail
5.	SET positionPan.0 to Mode: 3(Continuous), Speed:			Pass/Fail
	<conpanspeed>, Position: 0, which is hex value 03</conpanspeed>			
	<conpanspeed> 00 00</conpanspeed>			
6.	DELAY <alt_contpantimeout> milliseconds</alt_contpantimeout>			
7.	SET positionPan.0 to 00 00 00 00			Pass/Fail
8.	USER VERIFY the camera stops moving			Pass/Fail
9.	SET positionPan.0 to Mode: 3(Continuous), Speed: - Pass/Fail			Pass/Fail
	<conpanspeed>,</conpanspeed>	conPanSpeed>, Position: 0, which is hex value 03 -		
	<conpanspeed></conpanspeed>	• 00 00		
10.	DELAY <alt_cont< th=""><th><i>PanTimeout></i> milliseconds</th><th></th><th></th></alt_cont<>	<i>PanTimeout></i> milliseconds		
11.	SET positionPan.0 to 00 00 00 00		Pass/Fail	
12.	USER VERIFY the camera stops moving		Pass/Fail	
13.	SET timeoutPan.	0 = [timeoutPan]		Pass/Fail
Test Case Results				
Tested By:		Date		Pass/Fail
		Tested:		
Test Case Notes:				
Version History:	V1.0 09/20/05 Ir	nitial Draft – RDR		
-	V1.1 02/13/06 A	06 Added test for support of panning and pan limits – JJ		
	V1.2 02/22/06 Implemented script and proofed – JJ			

Change Administrator Community Name

Test Case:	Title:	Change Administrator Co	ommunity Name	
Security-TC0001	Description: This Test Case verifies that the administrator can change the			
····,		administrator community name stored in the DUT and that the		
	change properly affects the operation of the device.			vice.
	Variables:	0 1 1 3	·	
	Pass/Fail	The DUT shall pass every	uded within the	
	Criteria:	Test Case in order to pass	the Test Case.	
Test Step	Test Procedure			Results
Number				
1.	As a part of the pr	oject-specific Test Plan, def	ine a	
	<temp_communityname> to use for testing the administrator</temp_communityname>			
	community name.	This must be a different name	me than the	
	current communit	NameAdmin and community	tyNameUser(s)	
	values, which will	be referred to as the origina	Il community	
	name.			
	Nata Nalid admin	istrator community nomes o	re hetween 0 and	
	16 characters long	isitator community names a		
	characters long	g, inclusive and may contain	any ASCII	
2				
۷.	<pre>current commun</pre>	nitvName>		
3.	GET communityNameAdmin 0 = [communityNameAdmin]			Pass/Fail
4.	VERIFY RESPON	ISE VALUE= COMMUNITY	NAME OUT	Pass/Fail
5.	SET communityNameAdmin $0 = < temp communityNameN$			Pass/Fail
6.	GET communityNameAdmin.0 and VERIFY that either: Pass/Fail			Pass/Fail
	1. No response	e is received or		
	2. The RESP	SPONSE ERROR is 2 (noSuchName) and the		
	RESPONSE INDEX is 1			
7.	CONFIGURE COMMUNITY NAME OUT =			
	<temp_communit< td=""><td>yName></td><td></td><td></td></temp_communit<>	yName>		
8.	GET communityN	ameAdmin.0		Pass/Fail
9.	VERIFY RESPONSE VALUE = COMMUNITY NAME OUT			Pass/Fail
10.	SET communityNameAdmin.0 = [communityNameAdmin]			Pass/Fail
11.	GET communityNameAdmin.0 and VERIFY that either: Pass/Fail			Pass/Fail
	1. No respons	1. No response is received or		
		PONSE ERROR is 2 (noSuchName) and the		
10				
12.				
13	GET communityN	ameAdmin 0		Pass/Fail
10.	VERIEY RESPON	ISE VALUE = COMMUNITY	NAME OUT	Pass/Fail
	VERMITTEOLO	Test Case Results		1 400/1 411
Tested By:		Date		Pass/Fail
realed by:		Tested:		1 400/1 41
Test Case Notes:				
Version History:	V1.0 02/13/06 C	Priginal as defined in NTCIP	8007 v01.20 - B.3.4	Change
	Administra	ator Community Name – RD	R	
	V2.0 02/15/06 C	hanged "Change" to CONF	IGURE	
	Removed '	Temporary community nam	e" from Variables fie	ld and adopted
	convention of using [name of variable] for the text description of local			
	variables a	nd using <name> for extern</name>	ally defined constant	ts – RDR
	V2.1 02/22/06 li	mplemented script and proof	fed – JJ	

Change User Community Name

Test Case:	Title:	Change User Community Name		
Security-TC002	Description:	This Test Case verifies that the administrator can change the		
-		user community names and their masks stored in the DUT and		
		that the changes properly affect the operation of	f the device.	
	Variables:			
	Pass/Fail	The DUT shall pass every verification step inclu	ided within the	
	Criteria:	Test Case in order to pass the Test Case.		
Test Step	Test Procedure		Results	
Number				
1.	PRE-CONDITION	N: Ensure that the DUT is configured such that		
	the user commun	ity names are something other than "NTCIP		
	USER #", where #			
	table.			
	When set to 0, the	e communityNameAccessMask effectively		
	changes the any	object with read-write access to ready only.		
	The object to test	is <maskreadwriteoid>. The default value of</maskreadwriteoid>		
	the OID in the set	t up file is sysName.0. However, one may		
	replace these with	n any other object OID supported by the DUI		
	that that has simil	iar characteristics and is read-write.		
			Base/Fail	
۷.		lodules.0	Fa55/Faii	
	Note: This object	t is referenced throughout this procedure. A		
	Test Plan may re	place all references to this object in this Test		
	Case with a refer	ence to any other object supported by the DUT		
	that will have a co	onstant value for the duration of the Test Case.		
	The globalMaxMo	odules is used in this procedure because most		
	every device sup	ports it.		
3.	RECORD the RE	SPONSE VALUE = [globalMaxModules]		
4.	GET communityN	JamesMax.0	Pass/Fail	
5.	RECORD the RE	SPONSE VALUE = [communityNamesMax]		
6.	VERIFY RESPOR	NSE VALUE >= <req communitynamesmax=""></req>	Pass/Fail	
7.	FOR N = 1 TO [co	ommunityNamesMax]		
	-			
	Note: This loc	op tests the original community names, to make		
	sure that they	work and then SETs them to new values and		
	ensures that t	the new values work.		
8.	GET the follow	ving objects:	Pass/Fail	
	communityNa	ameUser.N,		
	communityNa	ameAccessMask.N		
9.	RECORD the	RESPONSE VALUES =		
	[communityNa	meUser(N)] and		
	[communityina	meAccessMask(N) for each object indicating		
10				
10.		melleer(N)]		
11	GET dobalMa	vModules 0	Pass/Fail	
11.			Dass/Fall	
13	GET cmaekDa	adWriteOID>	Pass/Fail	
1J.			F 033/F dii	
17.	Itemn mackDd	eadWriteOIDValue]		
15	SET <maskpa< th=""><th>advinteOIDvaluej</th><th>Dass/Eail</th></maskpa<>	advinteOIDvaluej	Dass/Eail	
10.			rass/raii	

	[temp_maskReadWriteOIDValue]		
	Note: This answers that community Name AssessMack Nie		
	Note: This ensures that communityNameAccessMask.N is		
16	CONFICURE COMMUNITY NAME OUT - administrator		
10.			
17	SET the following objects to the values as shown:	Dass/Eail	
17.	communityNameLiser N to the string "NTCIP LISER #"	r ass/r all	
	where $\#$ is the row number N		
	communityNameAccessMask N to the value of zero (0)		
18.	GET the following objects:	Pass/Fail	
	communityNameUser.N.		
	communityNameAccessMask.N		
19.	VERIFY RESPONSE VALUE	Pass/Fail	
-	communityNameUser.N = "NTCIP USER #"		
20.	VERIFY that the RESPONSE VALUE	Pass/Fail	
	communityNameAccessMask.N = 0		
21.	Change the COMMUNITY NAME OUT to "NTCIP USER #"		
	where # is the row number N		
22.	GET globalMaxModules.0	Pass/Fail	
23.	VERIFY RESPONSE VALUE = [globalMaxModules]	Pass/Fail	
24.	SET <maskreadwriteoid> = <maskreadwriteoidvalue></maskreadwriteoidvalue></maskreadwriteoid>	Pass/Fail	
	and VERIFY RESPONSE ERROR is 2 (noSuchName) and		
	the RESPONSE INDEX is 1		
	Note: This ensures that changing the access mask to 0 SETs		
	the access of <maskreadwriteoid> = readOnly.</maskreadwriteoid>		
25.	CONFIGURE the COMMUNITY NAME OUT to the		
	administrator community name		
26.			
27.	FOR N = 1 IO [communityNamesMax]		
	Note: This loop tests the original community names, to make		
28.	CONFIGURE COMMUNITY NAME OUT =		
20	CET risks/Madules 0 and VED/EV that sither	Dece/Feil	
29.	GET globalivaximodules.0 and VERIFY that either.	Pass/Fall	
	1. NO RESPONSE IS RECEIVED OF 2. The DESDONSE EDDOR is 2 (neSuchName) and the		
	2. THE RESPONSE ERROR IS 2 (HUSUCHNAILE) and the RESPONSE INDEX is 1		
30	NEXT N		
30.	FOR N = 1 TO [communityNamesMax]		
01.			
	Note: This loop tests the original community names once		
	restored to the DUT to make sure that they work.		
32.	CONFIGURE COMMUNITY NAME OUT to the <original< td=""><td></td></original<>		
	administrator community name		
33.	SET the following objects to the values as shown:	Pass/Fail	
	communityNameUser.N = [communityNameUser(N)],		
	communityNameAccessMask.N =		
	[communityNameAccessMask(N)]		
34.	GET the following objects:	Pass/Fail	
	communityNameUser.N,		
	communityNameAccessMask.N		

35.	VERIFY RESPONSE VALUE	Pass/Fail		
	communityNameUser.N = [communityNameUser(N)]			
36.	VERIFY RESPONSE VALUE	Pass/Fail		
	communityNameAccessMask.N =			
	[communityNameAccessMask(N)]			
37.	Change the COMMUNITY NAME OUT to the			
	[communityNameUser(N)]			
38.	GET globalMaxModules.0	Pass/Fail		
39.	VERIFY RESPONSE VALUE = [globalMaxModules]	Pass/Fail		
40.	NEXT N			
41.	FOR N = 1 TO [communityNamesMax]			
	Note: This loop tests the temperany community names			
	created during this Test Case to make sure that they no			
	longer work			
10				
42.	CONFIGURE COMMUNITY NAME OUT = NTCIP USER			
40	#", where # is the row number in			
43.		Pass/Fall		
	1. No response is received or			
	2. The RESPONSE ERROR is 2 (noSuchName) and the			
ΛΛ				
44. 15				
45.	configure community name out = administrator			
16	DECONDITION: Ensure that the DLIT is configured such that			
40.	PRE-CONDITION: Ensure that the DUT is configured such that			
	the user community names are something other than "NICIP			
	USER #, where # is the row number in the community name			
	ladie.			
	When not to 0, the community/NameAccess/Acck offectively			
	changes the any chiest with read write accessives to ready only			
	The object to toot is creat/Deed/WriteQUD>. The default value of			
	the OID in the actual file is available 0. However, one may			
	replace these with any other object OID supported by the DUT			
	that has similar characteristics and is read write			
	that has similar characteristics and is read-write.			
Test Case Results				
Tested By:	Date	Pass/Fail		
,	Tested:			
Test Case Notes:	If the Test Case fails, one or more of the user community names in	n the DUT may		
	be "NTCIP USER#", where # is the row number in the user commu	unity name table.		
Version History:	V1.0 02/13/06 Original as defined in NTCIP 8007 v01.20 - B.3.6	Change User		
,	Community Name	5		
	V2.0 02/15/06 Reformatted to use FOR and NEXT and simplified a number of			
	steps.			
	High-lighted potential invalid response to GET of globalMaxModules.0			
	Added last step to restore COMMUNITY NAME OUT			
	Add steps to check that setting of communityNameAccessMasks = 0			
	changes access to read-only.			
	Changed "Change" to CONFIGURE Removed "Temporary community name" from Variables field and adopted convention of using [name of variable] for the text description of local variables and using <name> for externally defined constants. – RDR</name>			
	V2.1 02/22/06 Implemented script and proofed – JJ			

Delta Tilt Motion

Test Case:	Title:	Delta Tilt Mo	tion		
Tilt-TC001	Description:	This Test Case tests the delta tilt motion of the camera by			
		moving the camera with several different speed and direction			
		parameters and allowing the user to verify them.			
	Variables:				
	Pass/Fail	The DUT sha	Il pass every	verification step inclu	uded within the
	Criteria:	Test Case in o	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	GET rangeTiltDov	wnLimit.0 and r	angeTiltUpLi	mit.0	Pass/Fail
2.	VERIFY camera	supports tilt limi	its		Pass/Fail
3.	SET positionTilt.0) to Mode: 1 (De	elta), Speed:		Pass/Fail
	<deltatitlmovesp< td=""><td>eed>, Position:</td><td><deltatiltmo< td=""><td><i>vement</i>>, which is</td><td></td></deltatiltmo<></td></deltatitlmovesp<>	eed>, Position:	<deltatiltmo< td=""><td><i>vement</i>>, which is</td><td></td></deltatiltmo<>	<i>vement</i> >, which is	
	hex value 01 < de	ltaTitlMoveSpe	ed> <deltati< td=""><td>iltMovement></td><td></td></deltati<>	iltMovement>	
4.	USER VERIFY the camera moved in a up direction at the Pass/Fail				
	movement and sp	and speed specified by <deltatiltmovespeed> and</deltatiltmovespeed>			
	<deltatiltmoveme< td=""><td colspan="4">Movement></td></deltatiltmoveme<>	Movement>			
5.	SET positionTilt.0) to Mode: 1 (De	elta), Speed:	-	Pass/Fail
	<deltatiltmovesp< td=""><td colspan="4">eltaTiltMoveSpeed>, Position: <deltatiltmovement>, which is</deltatiltmovement></td></deltatiltmovesp<>	eltaTiltMoveSpeed>, Position: <deltatiltmovement>, which is</deltatiltmovement>			
	hex value 01 - <de< td=""><td colspan="4">deltaTiltMoveSpeed> <deltatiltmovement></deltatiltmovement></td></de<>	deltaTiltMoveSpeed> <deltatiltmovement></deltatiltmovement>			
6.	USER VERIFY th	the camera moved in a down direction at the Pass/Fail			Pass/Fail
	movement and speed specified by <deltatiltmovespeed> and</deltatiltmovespeed>				
	<deltatiltmoveme< td=""><td>ent></td><td>-</td><td>-</td><td></td></deltatiltmoveme<>	ent>	-	-	
Test Case Results					
Tested By:			Date		Pass/Fail
-			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R		
-	V1.1 02/13/06 Added test for support of tilt limits – JJ				
	V1.2 02/22/06 Implemented script and proofed – JJ				
Absolute Tilt Motion

Test Case:	Title:	Absolute Til	t Motion	hsolute tilt motion of	the camera by
1111-1 6002	Description.	moving the c	amera with se	everal different speed	and direction
		parameters a	nd allowing th	ne user to verify them	n.
	Variables:		J	, ,	
	Pass/Fail	The DUT sha	II pass every	verification step inclu	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number				:4 0	D / [- 1]
1.		which interview which interview and realist the second s	ange HitupLii	mit.U	Pass/Fall
2.		supports tilt im	ILS haaluta) Sna	ad: 0. Desition: 0	Pass/Fall
э.	which is hex value	e 02 00 00 00	bsolute), Spe		Pass/Fall
4.	SET positionTilt.0) to Mode: 2 (A	bsolute), Spe	ed:	Pass/Fail
	<absolutetiltspe< td=""><td>ed>, Position: <</td><td><absolutetiltf< td=""><td>Position>, which is</td><td></td></absolutetiltf<></td></absolutetiltspe<>	ed>, Position: <	<absolutetiltf< td=""><td>Position>, which is</td><td></td></absolutetiltf<>	Position>, which is	
F		solute l'litSpee	d> <absolute< td=""><td>TIItPOSItion></td><td></td></absolute<>	TIItPOSItion>	
5.	USER VERIFY the camera moved to the position defined by <absolutetiltposition></absolutetiltposition>				Pass/Fail
6.	GET positionQue	GET positionQueryTilt.0			
7.	VERIFY RESPOR	NSE VALUE =	<absolutetiltl< td=""><td>Position></td><td>Pass/Fail</td></absolutetiltl<>	Position>	Pass/Fail
8.	SET positionTilt.0 which is hex value) to Mode: 2 (A e 02 00 00 00	bsolute), Spe	ed: 0, Position: 0,	Pass/Fail
9.	SET positionTilt.0 to Mode: 2 (Absolute), Speed:				Pass/Fail
	<absolutetiltspe< td=""><td></td></absolutetiltspe<>				
40	hex value 02 <ab< td=""><td>soluteTiltSpee</td><td>d> <absolute< td=""><td>TiltPosition2></td><td></td></absolute<></td></ab<>	soluteTiltSpee	d> <absolute< td=""><td>TiltPosition2></td><td></td></absolute<>	TiltPosition2>	
10.	<pre></pre>	ie camera mov ition2>	ed to the posi	ition defined by	Pass/Fail
11.	GET positionQue	ryTilt.0			Pass/Fail
12.	VERIFY RESPOR	NSE VALUE =	<absolutetiltl< td=""><td>Position2></td><td>Pass/Fail</td></absolutetiltl<>	Position2>	Pass/Fail
13.	SET positionTilt.0) to Mode: 2 (A	bsolute), Spe	ed: 0, Position: 0,	Pass/Fail
	which is hex valu	e 02 00 00 00			
	1	Test Case	Results	1	
Tested By:			Date Tested:		Pass/Fail
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD	R		
, j	V1.1 11/03/05 A	dded test for p	ositionQuery1	Filt — JJ	
	V1.2 02/03/06-A	Added step to s	et position to	Home (zero) before	testing absolute
	movement – JJ				
	V1.3 02/13/06 A	Added test for s	support of tilt I	imits – JJ	
	V1.4 02/22/06 I	mplemented so	cript and proo	fed – JJ	

Continuous Tilt Motion with Timeout

Test Case:	Title: Continuous Tilt Motion with Timeout						
Tilt-TC003	Description:	This Test Cas	se tests the c	ontinuous tilt motion	of the camera by		
		moving the camera with the continuous command using the					
	Mariahlaa	timeout parar	neter to stop	the camera.			
				varification atom inclu	ided within the		
	Critoria:	The DUT sha	order to pass	the Test Case			
Tost Stop	Test Procedure	Test Case III		the rest case.	Posults		
Number	rest Frocedure				Results		
1.	GET timeoutTilt.0	= [timeoutTilt]			Pass/Fail		
2.	SET timeoutTilt.0	to <alt_conttil< td=""><td>ltTimeout></td><td></td><td>Pass/Fail</td></alt_conttil<>	ltTimeout>		Pass/Fail		
3.	SET positionTilt.0	to Mode: 3 (C	ontinuous), S	peed:	Pass/Fail		
	<contiltspeed>, I</contiltspeed>	Position: 0, whi	ich is hex valι	le			
	03 <contiltspeed< td=""><td>/> 00 00</td><td></td><td></td><td></td></contiltspeed<>	/> 00 00					
4.	USER VERIFY th	e camera mov	es up and sto	ps after	Pass/Fail		
	<alt_conttilttime< td=""><td><i>out></i> millisecon</td><td>ds</td><td></td><td></td></alt_conttilttime<>	<i>out></i> millisecon	ds				
5.	SET positionTilt.0	to Mode: 3 (C	ontinuous), S	peed: -	Pass/Fail		
	<contiltspeed>, I</contiltspeed>	Position: 0, whi	ich is hex valu	le			
	03 - <contiltspeed< td=""><td>d> 00 00</td><td></td><td></td><td></td></contiltspeed<>	d> 00 00					
6.	USER VERIFY th	e camera mov	es down and	stops after	Pass/Fail		
_	<alt_conttilttime< td=""><td>out> millisecon</td><td>ds</td><td></td><td></td></alt_conttilttime<>	out> millisecon	ds				
7.	SET timeout lilt.0	= [timeout l ilt]			Pass/Fail		
		Test Case	Results				
Tested By:			Date		Pass/Fail		
-			Tested:				
Test Case Notes:							
Version History:	V1.0 09/20/05 In	iitial Draft – RD	R				
-	V1.1 02/22/06 In	nplemented sc	ript and proof	V1.1 02/22/06 Implemented script and proofed – JJ			

Continuous Tilt Motion with Stop

Test Case: Tilt-TC0004	<i>Title</i> : Description:	Continuous Tilt Motion with Stop This Test Case tests the continuous tilting motion of the camera by moving the camera and using the stop command to stop			
		movement.			
	Variables:				
	Pass/Fail	The DUT sha	ll pass every	verification step inclu	uded within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	GET timeoutTilt.0	= [timeoutTilt]			Pass/Fail
2.	SET timeoutTilt.0	to 0		-	Pass/Fail
3.	SET positionTilt.0	to Mode: 3 (Co	ontinuous), S	peed:	Pass/Fail
	<contiltspeed>,</contiltspeed>	Position: 0, whi	ch is nex valu	le	
	03 <con l="" litspeed<="" td=""><td>1>00.00 T:11:T:</td><td></td><td></td><td></td></con>	1>00.00 T:11:T:			
<u>4.</u>	DELAY <alt_cont lift="" limeout=""> milliseconds</alt_cont>			Dece/Feil	
5.	hex value 00 00 0	0 10 10100e: 0, 5p		lion: 0, which is	Pass/Fail
6.	USER VERIFY th	SER VERIFY the camera stops moving			Pass/Fail
7.	SET positionTilt.0 <contiltspeed>, I</contiltspeed>	to Mode: 3 (Co Position: 0, whi	ontinuous), S ch is hex valu	peed: - ue 03 -	Pass/Fail
•	<con l="" litspeed=""> 0</con>	10 00 Till(Time e e e e e			
8.	DELAY <ait_cont< td=""><td>Tillt Timeout> m</td><td></td><td>ion () which io</td><td></td></ait_cont<>	Tillt Timeout> m		ion () which io	
9.	hex value 00 00 0	0 10 10100e: 0, 5µ)0 00		lion: 0, which is	Pass/Fail
10.	USER VERIFY th	e camera stops	s moving		Pass/Fail
11.	SET timeoutTilt.0	= [timeoutTilt]			Pass/Fail
		Test Case	Results		
Tested By:			Date Tested:		Pass/Fail
Test Case Notes:				L	
Version History:	V1.0 09/20/05 Ir V1.1 02/22/06 In	nitial Draft – RD nplemented scr	R ipt and proof	ed – JJ	

Preset Position

Test Case:	Title:	Preset Position			
Zone-TC001	Description:	This Test Case stores and	I moves the camera t	o preset camera	
		positions.			
	Variables:				
	Pass/Fail	The DUT shall pass every	verification step inclu	uded within the	
	Criteria:	Test Case in order to pase	s the Test Case.		
Test Step	Test Procedure			Results	
Number					
1.	SET positionPan.	0 to 02 <presetmovements< th=""><th>peed></th><th>Pass/Fail</th></presetmovements<>	peed>	Pass/Fail	
	<presetpanpositi< th=""><th>on1> and positionTilt.0 to 02</th><th>2</th><th></th></presetpanpositi<>	on1> and positionTilt.0 to 02	2		
	<presetmovement< th=""><th>tSpeed> <presettiltposition< th=""><th>11></th><th></th></presettiltposition<></th></presetmovement<>	tSpeed> <presettiltposition< th=""><th>11></th><th></th></presettiltposition<>	11>		
2.	Note current cam	era position as position 1.			
3.	SET presetStore	Position.0 to <presetstore1:< th=""><th>></th><th>Pass/Fail</th></presetstore1:<>	>	Pass/Fail	
4.	SET positionPan.	0 to 02 <presetmovements< th=""><th>peed></th><th>Pass/Fail</th></presetmovements<>	peed>	Pass/Fail	
	<presetpanposition2> and positionTilt.0 to 02</presetpanposition2>				
	<presetmovement< th=""><th>tSpeed> <presettiltposition< th=""><th>า2></th><th></th></presettiltposition<></th></presetmovement<>	tSpeed> <presettiltposition< th=""><th>า2></th><th></th></presettiltposition<>	า2>		
5.	Note current cam	era position as position 2.			
6.	SET presetStore	SET presetStorePosition.0 to <presetstore2></presetstore2>			
7.	SET presetGotoF	SET presetGotoPosition.0 to <presetstore1></presetstore1>			
8.	USER VERIFY th	e camera moved to positior	<u>ו 1</u>	Pass/Fail	
9.	GET presetPositie	onQuery.0		Pass/Fail	
10.	VERIFY RESPOR	NSE VALUE = 1		Pass/Fail	
11.	SET presetGotoP	osition.0 to <presetstore2></presetstore2>	•	Pass/Fail	
12.	USER VERIFY th	e camera moved to positior	า 2	Pass/Fail	
13.	GET presetPositi	onQuery.0		Pass/Fail	
14.	VERIFY RESPON	NSE VALUE = 2		Pass/Fail	
	Test Case Results				
Tested By:		Date		Pass/Fail	
-		Tested:			
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RDR			
	V1.1 11/03/05 A	dded test for presetPosition	Query.0 – JJ		
	V1.2 02/27/06 Ir	nplemented script and proo	fed – JJ		

Get-Set Zone

Test Case:	Title:	Get-Set Zone	;		
Zone-TC002	Description:	This Test Cas	se tests the s	torage of camera zor	nes.
	Variables:				
	Pass/Fail	The DUT sha	ll pass every	verification step inclu	uded within the
	Criteria:	Test Case in o	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	GET zoneMaximu	um.0 = [zoneMa	aximum]		Pass/Fail
2.	VERIFY zoneMax	(imum.0 >= < <i>re</i>	eq_ZoneMaxi	mum>	Pass/Fail
3.	FOR zoneIndex =	1 to [zoneMax	imum]		
4.	GET				Pass/Fail
	zoneLabel.zone	eIndex = [zonela	ibel],		
	zonePanRightL	imit.zoneIndex =	= [zonePanRi	ghtLimit],	
	zoneTiltUpLimit	zoneIndex = [z	oneTiltUpLim	it],	
	zone HitDownLi	mit.zoneindex =	zone Hitbow	vnLimit],	
		hit.zoneindex =	[zonePanLeπ	Limitj	Decc/Foil
Э.	JEI Jonal abal Jona	unday = < alt =	nalabab		Pass/Fall
	zonePanPight	inuex – <i>Sall_2</i> 0	- < olt zonoD	onPightLimit	
		7000000000000000000000000000000000000	= <an_20nera< td=""><td>al imit></td><td></td></an_20nera<>	al imit>	
	zoneTiltDownl i	mit zonelndex =	∶ <alt td="" zonetil<=""><td>tDownl imit></td><td></td></alt>	tDownl imit>	
	zonePanLeftLim	hit.zoneIndex = -	<alt td="" zonepan<=""><td>LeftLimit></td><td></td></alt>	LeftLimit>	
6.	SET				Pass/Fail
	zoneLabel.zone	eIndex = [zonela	ibel],		
	zonePanRightL	imit.zoneIndex =	= [zonePanRi	ghtLimit],	
	zoneTiltUpLimit	.zoneIndex = [z	oneTiltUpLimi	it],	
	zoneTiltDownLi	mit.zoneIndex =	[zoneTiltDow	vnLimit],	
	zonePanLeftLim	nit.zoneIndex =	[zonePanLeft]	Limit]	
7.	NEXT zoneIndex				
Test Case Results					
Tested By:			Date		Pass/Fail
			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 In	iitial Draft – RD	R		
-	V1.1 02/27/06 In	nplemented scr	ript and proof	ed – JJ	

Move In and Out of Zone

Test Case: Zone-TC003	<i>Title</i> : Description:	Move In and Out of Zone This Test Case tests the labeling capability of zones by moving to areas within zones.		
	Variables: Pass/Fail Criteria:	The DUT shall pass every verification step inclu Test Case in order to pass the Test Case.	uded within the	
Test Step Number	Test Procedure		Results	
1.	GET zoneLabel.< <i>index.</i> zonePanRightLimi zoneTiltUpLimit.< <i>i</i> zoneTiltDownLimi	<i>Zone</i> > = [zoneLabel] it.< <i>indexZone</i> > = [zonePanRightLimit], <i>indexZone</i> > = [zoneTiltUpLimit], t.< <i>indexZone</i> > = [zoneTiltDownLimit],	Pass/Fail	

	zonePanLeftLimit. < <i>indexZone</i> > = [zonePanLeftLimit]	
2.	SET	Pass/Fail
	zoneLabel. < <i>indexZone</i> >= < <i>alt_zoneLabel</i> >,	
	zonePanRightLimit.< <i>indexZone</i> >= <alt_zonepanrightlimit>,</alt_zonepanrightlimit>	
	zoneTiltUpLimit.< <i>indexZone</i> >= <alt_zonetiltuplimit>,</alt_zonetiltuplimit>	
	zoneTiltDownLimit.< <i>indexZone</i> >= <alt_zonetiltdownlimit>,</alt_zonetiltdownlimit>	
	zonePanLeftLimit. <indexzone>= <alt_zonepanleftlimit></alt_zonepanleftlimit></indexzone>	
3.	GET	Pass/Fail
	labelText.< indexZone > = [labelText],	
	labelHeight.< indexZone > = [labelHeight],	
	labelColor.< IndexZone > = [labelColor],	
	labelStartColumn < indexZone > = [labelStartColumn]	
	abelStaticolumn. Muex20ne > = [labelStaticolumn], [abelStatus < index20ne > = [labelStatus]	
Λ		Pass/Fail
7.	lahelText < index7one > = <alt_7laheltext></alt_7laheltext>	1 833/1 81
	$ abe ext < index_one > = ,$	
	labelColor < indexZone > = <alt zlabelcolor=""></alt>	
	labelStartRow.< indexZone > = <alt zlabelstartrow="">.</alt>	
	labelStartColumn.< indexZone > = <alt_zlabelstartcolumn>,</alt_zlabelstartcolumn>	
	labelStatus.< indexZone > = <alt_zlabelstatus></alt_zlabelstatus>	
5.	SET	Pass/Fail
	positionPan.0 to 02 < zoneMoveSpeed> (<alt_zonerightlimit>-</alt_zonerightlimit>	
	1) and	
	positionTilt.0 to 02 < zoneMoveSpeed> (< alt_zoneUpLimit>-1)	
6.	USER VERIFY that the label for the current zone is correctly	Pass/Fail
	displayed, as set in step 2	
7.	SET	Pass/Fail
	positionPan.0 to 02 < <i>zoneMoveSpeed</i> >	
	(<alt_zonerightlimit>+1) and</alt_zonerightlimit>	
	positionTilt.0 to 02 < zoneMoveSpeed> (<alt_zoneuplimit>+1)</alt_zoneuplimit>	
8.	USER VERIFY that the label for the current zone is not	Pass/Fail
	displayed	
9.	SET	Pass/Fail
	zoneLabel.< <i>indexZone</i> > = [zoneLabel],	
	zonePanRightLimit.< <i>indexZone</i> > = [zonePanRightLimit],	
	zoneTiltUpLimit.< <i>indexZone</i> > = [zoneTiltUpLimit],	
	zone l iltDownLimit.< <i>indexZone</i> > = [zone l iltDownLimit],	
10	zoneranlettlimit. = [zoneranlettlimit]	
10.	JEI JohalTavt <i>c inday</i> Zana > = [JahalTavt]	Pass/Fall
	abelHeight < indexZone > = [labelHeight],	
	abel(log) = [abel(log)], abel(log) < index 20ne > = [label(log)]	
	abelStartRow < indexZone > = [labelStartRow]	
	labelStartColumn.< indexZone > = [labelStartColumn].	
	labelStatus.< indexZone > = [labelStatus]	
	Test Case Results	
Tested Bv:	Date	Pass/Fail
· · · · · · · · · · · · · · · · · · ·	Tested:	
Test Case Notes [.]		
Version History:	V1.0 09/20/05 Initial Draft – RDR	
	V1.1 11/03/05 Removed labelFontType. deprecated – JJ	
	V1.2 02/27/06 Implemented script and proofed – JJ	

Delta Zoom Motion

Test Case: Zoom-TC001	<i>Title</i> : Description:	Delta Zoom Motion This Test Case tests the delta zoom motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.			
	Variables: Pass/Fail Criteria:	The DUT sha Test Case in	Il pass every order to pass	verification step inclustion the Test Case.	uded within the
Test Step Number	Test Procedure				Results
1.	GET rangeZoom.	0			Pass/Fail
2.	VERIFY camera	supports zoom	limits		Pass/Fail
3.	SET positionZoor <deltazoommove which is hex value <deltazoommove< td=""><td>mLens.0 to Moo Speed>, Positi e 01 <deltazoo ement></deltazoo </td><td>de: 1(Delta), s on: <i><deltazo< i=""> <i>mMoveSpee</i></deltazo<></i></td><td>Speed: omMovement>, d></td><td>Pass/Fail</td></deltazoommove<></deltazoommove 	mLens.0 to Moo Speed>, Positi e 01 <deltazoo ement></deltazoo 	de: 1(Delta), s on: <i><deltazo< i=""> <i>mMoveSpee</i></deltazo<></i>	Speed: omMovement>, d>	Pass/Fail
4.	USER VERIFY the camera lens moved towards a telephoto position at the movement and speed specified by the test variables <deltazoommovespeed> and <deltazoommovement></deltazoommovement></deltazoommovespeed>				Pass/Fail
5.	SET positionZoor <deltazoommove which is hex value <deltazoommove< td=""><td>mLens.0 to Moo Speed>, Positi e 01 -<deltazo ement></deltazo </td><td>de: 1(Delta), s on: <i><deltazo< i=""> oomMoveSpe</deltazo<></i></td><td>Speed: - omMovement>, ed></td><td>Pass/Fail</td></deltazoommove<></deltazoommove 	mLens.0 to Moo Speed>, Positi e 01 - <deltazo ement></deltazo 	de: 1(Delta), s on: <i><deltazo< i=""> oomMoveSpe</deltazo<></i>	Speed: - omMovement>, ed>	Pass/Fail
6.	USER VERIFY the camera lens moved towards a wide angle position at the movement and speed specified by the test variables <deltazoommovespeed> and <deltazoommovement></deltazoommovement></deltazoommovespeed>			Pass/Fail	
		Test Case	Results		
Tested By:			Date Tested:		Pass/Fail
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir V1.1 02/13/06 A V1.2 02/27/06 Ir	nitial Draft – RD dded test for si nplemented sc	R upport of zoo ript and proof	m limits – JJ řed – JJ	

Absolute Zoom Motion

Test Case: Zoom-TC002	<i>Title</i> : Description: Variables: Pass/Fail Criteria:	Absolute Zoom Motion This Test Case tests the absolute zoom motion by moving the camera with several different spe parameters and allowing the user to verify them The DUT shall pass every verification step inclu Test Case in order to pass the Test Case.	of the camera eed and direction n. uded within the
Test Step	Test Procedure		Results
Number			
1.	GET rangeZoomt	.0	Pass/Fail
2.	VERIFY camera	supports zoom limits	Pass/Fail
3.	SET positionZoor	nLens.0 to 02 00 00 00	Pass/Fail
4.	SET positionZoor <absolutezoompo< td=""><td>nLens.0 to 02 <i><absolutezoomspeed></absolutezoomspeed></i> osition></td><td>Pass/Fail</td></absolutezoompo<>	nLens.0 to 02 <i><absolutezoomspeed></absolutezoomspeed></i> osition>	Pass/Fail
5.	VERIFY the came <absolutezoompe< td=""><td>era moved to the position defined by osition></td><td>Pass/Fail</td></absolutezoompe<>	era moved to the position defined by osition>	Pass/Fail

6.	GET positionQueryZoom.0		Pass/Fail		
7.	VERIFY RESPONSE VALUE =	Pass/Fail			
8.	SET positionZoomLens.0 to 02 (00 00 00		Pass/Fail	
9.	SET positionZoomLens.0 to 02 - <absolutezoomposition2></absolutezoomposition2>	<absolutezoo< td=""><td>mSpeed></td><td>Pass/Fail</td></absolutezoo<>	mSpeed>	Pass/Fail	
10.	VERIFY the camera moved to the position defined by <absolutezoomposition2></absolutezoomposition2>			Pass/Fail	
11.	GET positionQueryZoom.0			Pass/Fail	
12.	VERIFY RESPONSE VALUE = <absolutezoomposition2></absolutezoomposition2>			Pass/Fail	
Test Case Results					
Tested By:		Date Tested:		Pass/Fail	
Test Case Notes:					
Version History:	 V1.0 09/20/05 Initial Draft – RDR V1.1 11/03/05 Added tests for positionQueryZoom – JJ V1.2 02/13/06 Added steps to set position to Home position to test absolute movements. Added test for support of zoom limits – JJ V1.3 02/27/06 Implemented script and proofed – JJ 				

Continuous Zoom Motion with Timeout

Test Case:	Title:	Continuous Zoom Motion with Timeout			
Zoom-TC003	Description:	This Test Cas	se tests the c	ontinuous zoom moti	on of the camera
		by moving the	e camera with	the continuous com	mand using the
		timeout parar	neter to stop	the camera.	
	Variables:				
	Pass/Fail	The DUT sha	Ill pass every	verification step inclu	ided within the
	Criteria:	Test Case in	order to pass	the Test Case.	
Test Step	Test Procedure				Results
Number					
1.	GET timeoutZoor	n.0 = [timeoutZ	loom]		Pass/Fail
2.	SET timeoutZoom.0 to <alt_contzoomtimeout></alt_contzoomtimeout>			Pass/Fail	
3.	SET positionZoomLens.0 to 03 <conzoomspeed> 00 00</conzoomspeed>			Pass/Fail	
4.	USER VERIFY th	e camera lens	stops moving	g towards a	Pass/Fail
	telephoto position	n after <alt_con< td=""><td>ntZoomTimeo</td><td>ut> milliseconds</td><td></td></alt_con<>	ntZoomTimeo	ut> milliseconds	
5.	SET positionZoor	nLens.0 to 03 -	- <conzoomsp< td=""><td>peed> 00 00</td><td>Pass/Fail</td></conzoomsp<>	peed> 00 00	Pass/Fail
6.	USER VERIFY th	e camera lens	stops moving	g towards a wide	Pass/Fail
	angle position after	er <alt_contzoo< td=""><td>omTimeout></td><td>milliseconds</td><td></td></alt_contzoo<>	omTimeout>	milliseconds	
7.	SET timeoutZoon	n.0 = [timeoutz	Zoom]		Pass/Fail
		Test Case	Results		
Tested By:			Date		Pass/Fail
			Tested:		
Test Case Notes:					
Version History:	V1.0 09/20/05 Ir	nitial Draft – RD)r		
	V1.1 02/27/06 In	nplemented sc	ript and proof	ed – JJ	

Continuous Zoom Motion with Stop

Test Case:	Title:	Continuous Zoom Motior	with Stop			
Zoom-TC004	Description:	This Test Case tests the co	ontinuous zooming m	notion of the		
		camera by moving the camera and using the stop command to				
		stop movement.				
	Variables:					
	Pass/Fail	The DUT shall pass every	verification step inclu	uded within the		
	Criteria:	Test Case in order to pass	the Test Case.			
Test Step	Test Procedure			Results		
Number						
1.	GET timeoutZoon	n.0 = [timeoutZoom]		Pass/Fail		
2.	SET timeoutZoon	n.0 to 0		Pass/Fail		
3.	SET positionZoor	SET positionZoomLens.0 to 03 <conzoomspeed> 00 00</conzoomspeed>				
4.	DELAY <alt_cont< td=""><td>ZoomTimeout> milliseconds</td><td></td><td></td></alt_cont<>	ZoomTimeout> milliseconds				
5.	SET positionZoor	nLens.0 to 00 00 00 00		Pass/Fail		
6.	USER VERIFY th	e camera stops moving		Pass/Fail		
7.	SET positionZoor	mLens.0 to 03 - <i><conzoomsp< i=""></conzoomsp<></i>	oeed> 00 00	Pass/Fail		
8.	DELAY <alt_cont< td=""><td>ZoomTimeout> milliseconds</td><td></td><td></td></alt_cont<>	ZoomTimeout> milliseconds				
9.	SET positionZoor	mLens.0 to 00 00 00 00		Pass/Fail		
10.	USER VERIFY th	e camera stops moving		Pass/Fail		
11.	SET timeoutZoon	n.0= [timeoutZoom]		Pass/Fail		
		Test Case Results				
Tested By:		Date		Pass/Fail		
		Tested:				
Test Case Notes:						
Version History:	V1.0 09/20/05 Ir	nitial Draft – RDR				
-	V1.1 02/27/06 In	nplemented script and proofe	ed – JJ			

REFERENCES FOR APPENDIX C

- NTCIP 1205 Object Definitions for Closed Circuit Television (CCTV) Camera Control, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standard=1</u> <u>205</u>. Accessed August 17, 2006.
- NTCIP 1201 Global Object Definitions, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standard=1</u> 201. Accessed August 17, 2006.
- Test Procedures for NTCIP-conformant Closed Circuit Television (CCTV) Camera Controllers, Enterprise Consortium. <u>http://enterprise.prog.org/</u> (document no longer available) Accessed December 2002.
- 4. NTCIP 8007 Testing and Conformity Assessment Documentation within NTCIP Standards Publications, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standard=8</u>007. Accessed August 17, 2006.

APPENDIX D: CCTV PROTOCOL IMPLEMENTATION CONFORMANCE SPECIFICATION

INTRODUCTION

This appendix contains a Protocol Implementation Conformance Specification (PICS) for a specific camera and controller that was tested using the Prequalification Test Case - TC001 appearing in Appendix C. By adding indications that an implementation has support for a feature, an NTCIP Profile Requirements List (PRL) becomes a PICS. In the context of NTCIP testing, the PRL provides a standardized format to record test results that check for object support, maximum or limit values, index values, and the range of supported values.

NTCIP 1205 - Object Definitions for Closed Circuit Television (CCTV) Camera Control (NTCIP 1205-CCTV) dictates the format of the PICS (1). NTCIP 1205-CCTV grants specific copyright permission to use Annex B of that standard for creating a PICS (1). Except for the object support and supported values column entries, the format comes from the original PRL as it appears in the standard. The researcher indicate changes made in Amendment 1 v10 versus NTCIP 1205 v01.08 by highlighting them in bold-italics (1).

INTERPRETING RESULTS

For use in reporting test results, the words in all uppercase letters in the object support column indicate the results of specific object tests. A "YES" in the support column indicates that a retrieval operation did not return an error and, therefore, has support. Words appearing in mixed case in the object support column (for example, "Yes" or "Yes / No") indicate the absence of any specific test and appear as in the original. A single value in the supported values column (for example, "64") indicates a constant value supplied by the implementation. A set of passed or failed values in the supported values column (for example, "PASSED: 0, 1, FAILED: -1," indicates the implementation's responses to test values that were sent to the object. The test script that implements the Prequalification Test Case records the results directly onto the form.

Annex B INFORMATION PROFILE

(Informative)¹

Notice – PRL excerpted from a draft document containing preliminary information that is subject to change. Object names in bold-italics have additional information indicating changes between Version 01.08 and Amendment 1.

A Conformance Group is a basic unit of conformance and is used to specify a collection of related managed objects. The Conformance Group designation applied to a set of objects provides a systematic means for determining which objects are required to support a function. If a device has multiple functions, a Conformance Group will be defined for each function. Conformance Group definitions will be found in the NTCIP Object Definition Standard documents. The Object Definition Standard may define a Conformance Group with objects that are not in lexicographic order and only apply to devices of that type.

The related managed objects of a Conformance Group may include mandatory and/or optional objects. Mandatory objects within a Conformance Group shall be implemented. Optional objects shall be implemented only if a defined function of the device requires that particular object.

For example, assume a device implements an asynchronous RS-232 interface. It must implement all the mandatory objects in the Asynchronous Conformance Group of the RS-232 MIB. It would not have to implement the Synchronous Conformance Group of objects unless it also provided a synchronous interface.

Assume also that the Asynchronous Conformance Group has a *CRC error counter* object that is optional. The *CRC error counter* object would not have to be implemented unless the device used CRC checking on the asynchronous interface.

Conformance Groups are defined as either mandatory or optional. If a Conformance Group is mandatory, all of the objects with STATUS "mandatory" that are part of the Conformance Group shall be present for a device to claim conformance to the Conformance Group. If a Conformance Group is optional, all of the objects that are part of the Conformance Group with the STATUS "mandatory" shall be present if the device supports the Conformance Group. Objects with the STATUS "optional" may be supported.

When a table is included in a Conformance Group, all objects contained in the table are included by reference. This is because a table is defined as a SEQUENCE OF {SEQUENCE}. Thus, all objects listed in the sequence are defined as an integral part of the table. Tables are defined as either mandatory or optional. If a table is mandatory, all of the objects with STATUS "mandatory" shall be present. If a table is optional, all of the objects with the STATUS "mandatory" shall be present if the device supports the table. Objects in the table with the STATUS "optional" may be supported.

¹ Copyright release of Annex B Information Profile: Users of this standard may freely reproduce this Annex so that it may be used for its intended purpose and may further publish the completed Information Profile.

Based on NTCIP 1205:2001 v01.08 Amendment 1 v10 PRL Page B-2

B.1 NOTATION

The following notations and symbols are used to indicate status and conditional status within this standard.

B.1.1 TYPE Symbols

The following symbols are used to indicate type:

Symbol	Туре
С	Control Object - use of 'dbCreateTransaction' in NTCIP 1201 Clause 2.3.1 shall NOT delay a SET to this object.
Р	Parameter Object - use of 'dbCreateTransaction' in NTCIP 1201 Clause 2.3.1 to SET this object is optional.
P2	Parameter Object - use of 'dbCreateTransaction' in NTCIP 1201 Clause 2.3.1 to SET this object is mandatory.
S	Status / Information Object - this object is read only therefore a SET is not permitted.

B.1.2 Status Symbols

The following symbols are used to indicate status:

Symbol	Status
M	Mandatory
M. <n></n>	Support of every item of the group labeled by the same numeral <n> required, but only one is active at time.</n>
0	Optional
0. <n></n>	Optional, but support of at least one of the group of options labeled by the same numeral <n> is required</n>
С	Conditional
D	Deprecated
N/A	Non-applicable (i.e., logically impossible in the scope of the profile)
Х	Excluded or prohibited

B.1.3 Conditional Status Notation

The following predicate notation is used:

Notation	Status
<predicate>: M</predicate>	Item is conditional on the <predicate>.</predicate>

The <predicate>: notation means that the Status following it applies only when the feature or features identified by the predicate are supported. In the simplest case, <predicate> is the identifying tag of a single item.

B.1.4 Support Column

This section is in the form of a PICS and, therefore, includes a support column. An implementer claims support of an item by circling the appropriate answer (Yes or No) in the support column:

Original text ©2001 - 2005 AASHTO / ITE / NEMA

B.2 CCTV CAMERA CONTROL REQUIREMENTS

The Conformance Group definitions for CCTV Camera Control devices are defined in this clause. A CCTV Switch has multiple functions; thus, Conformance Groups are defined for each function.

The following table lists functional requirements for a CCTV Camera Control device, and asks if the listed features have been implemented.

Ref	Areas	Clause of Profile	Status	Support
B.3	CCTV Configuration Conformance Group	NTCIP 1205 – 3.2, 3.3 and 3.11	М	YES
B.4	CCTV Extended Functions Conformance Group	NTCIP 1205 – 3.6, 3.7, 3.8, 3.9 and 3.10	0	YES
B.5	CCTV Motion Control Conformance Group	NTCIP 1205 – 3.4 and 3.5	0	YES
B.6	CCTV On-Screen Menu Control	NTCIP 1205 – 3.12	0	YES
B.7	Configuration Conformance Group	NTCIP 1201 v01, Amendment 1	М	Yes
B.8	NTCIP Security Conformance Group	NTCIP 1201 v01, Amendment 1	М	Yes

CCTV Camera Control devices shall adhere to the conformance requirements specified in the above table as a minimum to claim compliance to this standard. Additional objects or groups may be supported without being non-compliant with CCTV Camera Control objects or NTCIP.

Minimum and maximum ranges of objects that differ from the values of the object's SYNTAX field may be enforced by an application running on a device.

A device which enforces range limits within the bounds specified by the values of the object's SYNTAX field shall not be categorized as being non-compliant with CCTV Camera Control objects or NTCIP.

A device which supports a subset of objects with enumerated values shall not be categorized as being non-compliant with CCTV Camera Control objects or NTCIP.

B.3 CCTV CONFIGURATION CONFORMANCE GROUP

The CCTV Configuration Conformance Group consists of the following objects:

	CCTV Configuration CO	NFORMAN	ICE GROUP			
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
3.2, 3.3 and 3.11	CCTV Configuration Conformance Group		М	YES		
3.2	CCTV Range Objects					
3.2.1	rangeMaximumPreset	S	3.2 : M	YES	0-255	64
3.2.2	rangePanLeftLimit	S	3.2 : M	YES	0-35999	35999
3.2.3	rangePanRightLimit	S	3.2 : M	YES	0-35999 65535	35999
3.2.4	rangePanHomePosition	S	3.2 : M	YES	0-35999 65535	0
3.2.5	rangeTrueNorthOffset	Р	3.2 : M	YES	0-35999 65535	PASSED: 0,1,18000, 35998,359 99,36000, FAILED: - 1,
3.2.6	rangeTiltUpLimit	S	3.2 : M	YES	0-35999 65535	1500
3.2.7	rangeTiltDownLimit	S	3.2 : M	YES	0-35999 65535	27000
3.2.8	rangeZoomLimit	S	3.2 : M	YES	0-65535	65535
3.2.9	rangeFocusLimit	S	3.2 : M	YES	0-65535	0
3.2.10	rangeIrisLimit	S	3.2 : M	YES	0-65535	0
3.2.11	rangeMinimumPanStepAngle	S	3.2 : M	YES	65535	10
3.2.12	rangeMinimumTiltStepAngle	S	3.2 : M	YES	0-35999 65535	10
3.3	CCTV Timeout Objects					
3.3.1	timeoutPan	Р	3.3 : M	YES	0-65535	PASSED: 1, FAILED: - 1,0,32767, 65535,655 36,
3.3.2	timeoutTilt	Р	3.3 : M	YES	0-65535	PASSED: 1, FAILED: - 1,0,32767, 65535,655 36.
3.3.3	timeoutZoom	Р	3.3 : M	YES	0-65535	PASSED: 1, FAILED: - 1,0,32767, 65535,655 36,
3.3.4	timeoutFocus	Ρ	3.3 : M	YES	0-65535	PASSED: 1, FAILED: - 1,0,32767, 65535,655 36,
3.3.5	timeoutIris	Р	3.3 : M	YES	0-65535	PASSED: 1, FAILED: - 1,0,32767, 65535.655

Original text 02001-2005 AASHTO / ITE / NEMA

Used by Permission

	CCTV Configuration CO	NFORMAN	ICE GROUP			
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
						36,
3.11	CCTV Label Objects					
3.11.1		<u> </u>	3.11.1VI 3.11 · M	VES	0-255	80
0.11.2	labelFntry		3.11 : M	YES		
3.11.2.1	labelIndex	S	3.11 : M	YES	0-255	
3.11.2.2	labelText	Р	3.11 : M	YES	String	PASSED: "","012345 6789ABCD EFGHIJ"," KLMNOPQ RSTUVWX YZ:./",
3.11.2.3	labelFontType (Deprecated in Amend v1.08)	P	3.11 : D	YES	0-255	
3.11.2.4	labelHeight	P	3.11 : M	YES	0-255	PASSED: 256, FAILED: 0,1,127,25 5,
3.11.2.5	labelColor	Р	3.11 : M	YES	1-16	PASSED: 0,7,17,
	blue(1)			Yes / No		
	green(2)			Yes / No		
	red(4)			Yes / No		
	magenta(5)			Yes / No		
	brown(6)			Yes / No		
	white(7)			YES		
	grey(8)			Yes / No		
	lightBlue(9)			Yes / No		
	lightGreen(10)			Yes / No		
	lightCyan(11)			Yes / No		
	lightRed(12)			Yes / No		
	lightMagenta(13)			Yes / No		
	yellow(14)			Yes / No		
	block(16)					
3.11.2.6	labelStartRow	P	3.11 : M	YES	0-255	PASSED: - 1,1,256, FAILED: 0,127,254, 255,
3.11.2.7	labelStartColumn	Р	3.11 : M	YES	0-255	PASSED: - 1,1,256, FAILED: 0,127,254, 255,
3.11.2.8	labelStatus	S	3.11 : M	YES	String	
	bit 7 – Label is Valid for Display			Yes		
	bit 6 – Display Status of Label			Yes		
	bit 5 – Reserved					
	DII 4 - Keserved					
	DII 3 - Keserved					
	bit 1 – Reserved					
	bit 0 – Reserved					
3.11.2.9	labelActive (Added in Amend v1.08)	P	3.11 : M	YES	String	PASSED: 0x00,0x80, 0x7F,0x00 00,
	bit 7 – Display Label			Yes		
l	bit 6 – Reserved					

Based on NTCIP 1205:2001 v01.08 Amendment 1 v10 PRL Page B-6

	CCTV Configuration CONFORMANCE GROUP					
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
	bit 5 – Reserved					
	bit 4 – Reserved					
	bit 3 – Reserved					
	bit 2 – Reserved					
	bit 1 – Reserved					
	bit 0 – Reserved					
3.11.2.9	labelFontNumber (Added in Amend v1.08)	Р	3.11 : M	YES	1-255	PASSED: 0,1,2,3,254 ,255,256,
3.11.3	labelLocationLabel	Р	3.11 : M	YES	0-255	PASSED: - 1,0,1,2,255 , FAILED: 256,
3.11.4	labelEnableTextDisplay	Ρ	3.11 : M	YES	String	PASSED: 0x00,0x80, 0x0000, FAILED: 0x7F,
	bit 7 – Display All Labels at Once			YES		
	bit 6 – Reserved					
	bit 5 – Reserved					
	bit 4 – Reserved					
	bit 3 – Reserved					
	bit 2 – Reserved					
	bit 1 – Reserved					
	bit 0 – Reserved					

B.4 CCTV EXTENDED FUNCTIONS CONFORMANCE GROUP

	CCTV Extended Functions CONFORMANCE GROUP					
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
3.6, 3.7, 3.8, 3.9 and 3.10	CCTV Extended Functions Conformance Group		0	YES		
3.6	CCTV System Feature Control Objects					
3.6.1	systemCameraFeatureControl	с	3.6 : M	YES	String	PASSED: 0x8000,0x 0080, FAILED: 0x4000,0x 2000,0x10 00,0x0800, 0x037F,
	Byte 1, bit 7 – Camera Power Select			Yes		
	Byte 1, bit 6 – Heater Power Select			Yes		
	Byte 1, bit 5 – Wiper Select			Yes		
	Byte 1, bit 4 – Washer Select			Yes		
	Byte 1, bit 3 – Blower Select			Yes		
	Byte 1, bit 2 – Reserved					
	Byte 1, bit 1 – Reserved					
	Byte 1, bit 0 – Reserved					
	Byte 2, bit 7 – Activation and Deactivation of the Camera Component			Yes		
	Byte 2, bit 6 – Reserved					
	Byte 2, bit 5 – Reserved					
	Byte 2, bit 4 – Reserved					
	Byte 2, bit 3 – Reserved					
	Byte 2, bit 2 – Reserved					
	Byte 2, bit 1 – Reserved					
	Byte 2, bit 0 – Reserved					
3.0.2	systemCameraFeatureStatus	5	3.6 : 17	YES	String	
	bit 6 Leaster Dewer Status			Yes		
	bit 5 - Minor Status			Vee		
	bit 4 Washer Status			Voc		
	bit 2 Blower Status			Voc		
	bit 2 Poconvod			165		
	bit 1 – Reserved					
	hit 0 – Reserved					
363	systemCameraEquipped	S	36 · M	YES	String	0x80
0.0.0	bit 7 – Camera Power Available			Yes		
	bit 6 – Heater Power Available			Yes		
	bit 5 – Wiper Available			Yes		
	bit 4 – Washer Available			Yes		
	bit 3 – Blower Available			Yes		
	bit 2 – Reserved					
	bit 1 – Reserved					
	bit 0 – Reserved					
3.6.4	systemLensFeatureControl	с	3.6 : M	YES	String	PASSED: 0x0000,0x 8000,0x40 00,0x0080, FAILED: 0x3F7F,
	Byte 1, bit 7 – Auto Iris Select			Yes		
	Byte 1, bit 6 – Auto Focus Select			Yes		
	Byte 1, bit 5 – Reserved					
	Byte 1, bit 4 – Reserved					
	Byte 1, bit 3 – Reserved					
	Byte 1, bit 2 – Reserved					

The CCTV Extended Functions Conformance Group consists of the following objects:

	CCTV Extended Functions	CONFORMA	NCE GROU	Р		
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
	Byte 1, bit 1 – Reserved					
	Byte 1, bit 0 – Reserved					
	Byte 2, bit 7 – Activation and]		Yes		
	Deactivation of the Lens Component	_		100		
	Byte 2, bit 6 – Reserved					
	Byte 2, bit 5 – Reserved					
	Byte 2, bit 4 – Reserved					
	Byte 2, bit 3 – Reserved					
	Byte 2, bit 2 – Reserved					
	Byte 2, bit 1 – Reserved					
0.05	Byte 2, bit U – Reserved				 Otring	
3.6.5	SystemLensFeatureStatus	5	3.6 : IVI	YES	String	
				Yes		
	DIT 6 - AUTO FOCUS Status			res		
	bit 2 Decenved					
	bit 2 Decenved					
	bit 1 Bosonyod					
	bit 0 Received					
366	DILU - RESERVEU		36 · M	 VES	String	
3.0.0	bit $7 = Auto Iris Available$		3.0 . ivi		Sung	0,00
	bit 6 Auto Eccus Available			Voc		
	bit 5 Decenved			165		
	bit 4 Decenved					
	bit 2 Decenved					
	bit 2 Decenved					
	bit 0 Beconved					
37						
3. /			27 · M		 String	
3.1.1	hit 7 Cohinet Alarm Status	3	3.7 . IVI	I LO Ves	Sting	
	bit 6 - Enclosure Alarm Status			Ves		
	bit 5 – Video Loss Alarm Status			Yes		
	bit 4 – Temperature Alarm Status			Yes		
	hit 3 – Pressure Alarm Status			Yes		
	bit 2 – Local/Remote Alarm Status			Yes		
	hit 1 – Washer Fluid Alarm Status			Yes		
	hit 0 - Reserved					
372	alarml atchStatus	s	37:M	YES	String	
<u> </u>	hit 7 – Cabinet Alarm Latch Status			Yes		
	bit 6 – Enclosure Alarm Latch Status	·		Yes		
	bit 5 – Video Loss Alarm Latch Status			Yes		
	bit 4 – Temperature Alarm Latch Status			Yes		
	bit 3 – Pressure Alarm Latch Status			Yes		
	bit 2 – Local/Remote Alarm Latch Status			Yes		
	bit 1 – Washer Fluid Alarm Latch Status			Yes		
	bit 0 – Reserved					
3.7.3	alarmLatchClear	Ρ	3.7 : M	YES	String	PASSED: 0x00,0x80, 0x40,0x20, 0x10,0x08, 0x04,0x02, FAILED: 0x01,
	bit 7 – Cabinet Alarm Latch Clear			Yes		
	bit 6 – Enclosure Alarm Latch Clear			Yes		
	bit 5 – Video Loss Alarm Latch Clear			Yes		
	bit 4 – Temperature Alarm Latch Clear			Yes		
	bit 3 – Pressure Alarm Latch Clear			Yes		
	bit 2 – Local/Remote Alarm Latch Clear			Yes		

CCTV Extended Functions CONFORMANCE GROUP						
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
	bit 1 – Washer Fluid Alarm Latch Clear			Yes		
3.7.4	bit 0 – Reserved alarmTemperatureHighLowThreshold	 P	 3.7 : M	YES	 String	 PASSED: 0x0000,0x 0028,0x80 7F,0x0000 00,
	Byte 1 – Low Temperature Threshold			Yes		
	Byte 2 – High Temperature Threshold			Yes		
3.7.5	alarmTemperatureCurrentValue	S	3.7 : M	YES	String	
	Byte 1 – Current Temperature Value			Yes		
3.7.6	alarmPressurehighLowThreshold	Ρ	3.7 : M	YES	String	PASSED: 0x0000,0x 0528,0x00 14,0x0000 00,
	Byte 1 – Low Pressure Threshold			Yes		
	Byte 2 – High Pressure Threshold			Yes		
3.7.7	alarmPressureCurrentValue	S	3.7 : M	YES	String	
	Byte 1 – Current Pressure Value			Yes		
3.7.8	alarmWasherFluidHighLowThreshold	Р	3.7 : M	YES	String	PASSED: 0x0000,0x 0A5A,0x05 55,0x0000 00,
	Byte 1 – Low Washer Fluid Threshold			Yes		
	Byte 2 – High Washer Fluid Threshold			Yes		
3.7.9	alarmWasherFluidCurrentValue	S	3.7 : M	YES	String	
	Byte 1 – Current Washer Fluid Value			Yes		
3.7.10	alarmLabelIndex	Ρ		YES		FAILED: 0x0000000 000000,0 x01020304 050607,0x 070605040 30201,0xF FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFFFF
	Byte 1 – Cabinet Alarm Label Number			Yes		
	Byte 2 – Enclosure Alarm Label Number			Yes		
	Byte 3 – Video Loss Alarm Label Number			Yes		
	Byte 4 – Temperature Alarm Label Number			Yes		
	Byte 5 – Pressure Alarm Label Number			Yes		
	Byte 6 – Local/Remote Alarm Label Number			Yes		
	Byte 7 – Washer Fluid Alarm Label Number			Yes		
3.7.11	alarmLabelSource (Added in Amend v1.08)	Р	3.7 : M	YES	String	PASSED: 0x00,0x54, 0xFF,0x00 00,
	bit 7 – Cabinet Alarm Latch Status			Yes		
	bit 6 – Enclosure Alarm Latch Status			Yes		
	bit 5 – Video Loss Alarm Latch Status			Yes		
	bit 4 – Temperature Alarm Latch Status			Yes		
	bit 3 – Pressure Alarm Latch Status			Yes		
	bit 2 – Local/Remote Alarm Latch Status			Yes		
	bit 1 – Washer Fluid Alarm Latch Status			Yes		
	bit 0 – Reserved					
3.8	CCTV Discrete Input Objects					
3.8.1	inputStatus	S	3.8 : M	YES	String	
	bit 7 – Discrete Input 8 Active Status			Yes		

	CCTV Extended Functions CONFORMANCE GROUP					
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
	bit 6 – Discrete Input 7 Active Status			Yes		
	bit 5 – Discrete Input 6 Active Status			Yes		
	bit 4 – Discrete Input 5 Active Status			Yes		
	bit 3 – Discrete Input 4 Active Status			Yes		
	bit 2 – Discrete Input 3 Active Status			Yes		
	bit 1 – Discrete Input 2 Active Status			Yes		
	bit 0 – Discrete Input 1 Active Status			Yes		
3.8.2	inputLatchStatus	S	3.8 : M	YES	String	
	bit 7 – Discrete Input 8 Latch Status			Yes		
	bit 6 – Discrete Input 7 Latch Status			Yes		
	bit 5 – Discrete Input 6 Latch Status			Yes		
	bit 4 – Discrete Input 5 Latch Status			Yes		
	bit 3 – Discrete Input 4 Latch Status			Yes		
	bit 2 – Discrete Input 3 Latch Status			Yes		
	bit 1 – Discrete Input 2 Latch Status			Yes		
	bit 0 – Discrete Input 1 Latch Status			Yes		
3.8.3	inputLatchClear	С	3.8 : M	YES	String	PASSED: 0x00,0xFF, 0x55,0x00 00,
	bit 7 – Discrete Input 8 Latch Clear			Yes		
	bit 6 – Discrete Input 7 Latch Clear			Yes		
	bit 5 – Discrete Input 6 Latch Clear			Yes		
	bit 4 – Discrete Input 5 Latch Clear			Yes		
	bit 3 – Discrete Input 4 Latch Clear			Yes		
	bit 2 – Discrete Input 3 Latch Clear			Yes		
	bit 1 – Discrete Input 2 Latch Clear			Yes		
	bit 0 – Discrete Input 1 Latch Clear			Yes		
3.8.4	inputLabelIndex	Ρ	3.8 : M	YES	String	FAILED: 0x000000, 0x010203, 0xFFFFFF, 0x0102030 4.
	Bvte 1 – Discrete Input 1 Label Number			Yes		
	Byte 2 – Discrete Input 2 Label Number			Yes		
	Byte 3 – Discrete Input 3 Label Number			Yes		
	Byte 4 – Discrete Input 4 Label Number			Yes		
	Byte 5 – Discrete Input 5 Label Number			Yes		
	Byte 6 – Discrete Input 6 Label Number			Yes		
	Byte 7 – Discrete Input 7 Label Number			Yes		
	Byte 8 – Discrete Input 8 Label Number			Yes		
3.8.5	inputPresetIndex (Added in Amend v1.08)	Ρ	3.8 : M	YES	String	PASSED: 0x000000 00000000 0x010203 040506070 8,0x08070 605040302 01,0xFFFF FFFFFFF FFFFFFFFFFFFFFFFFFFFFFFF
	Byte 1 – Discrete Input 1 Preset Number			Yes		
	Byte 2 – Discrete Input 2 Preset Number			Yes		
	Byte 3 – Discrete Input 3 Preset Number			Yes		
	Byte 4 – Discrete Input 4 Preset Number			Yes		
	Byte 5 – Discrete Input 5 Preset Number			Yes		
	Byte 6 – Discrete Input 6 Preset Number			Yes		
	Byte 7 – Discrete Input 7 Preset Number			Yes		
	Byte 8 – Discrete Input 8 Preset Number			Yes		
3.8.6	inputLabelSource (Added in Amend v1.08)	Р	3.8 : M	YES	String	PASSED: 0x00,0xFF,

Used by Permission

CCTV Extended Functions CONFORMANCE GROUP						
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
						0x0000,
	bit 7 – Discrete Input 8 Label Source			Yes		
	bit 6 – Discrete Input 7 Label Source			Yes		
	bit 5 – Discrete Input 6 Label Source			Yes		
	bit 4 – Discrete Input 5 Label Source			Yes		
	bit 3 – Discrete Input 4 Label Source			Yes		
	bit 2 – Discrete Input 3 Label Source			Yes		
	bit 0 Discrete Input 1 Label Source			Yes		
2.0	CCTV Discrete Output Objects			Tes		
3.9		 e	 2 Q · M	 VES	String	
3.9.1	bit 7 – Discrete Output 8 Active Status		3.9 . IVI	TES Vec	String	
	bit 6 – Discrete Output 7 Active Status			Ves		
	bit 5 – Discrete Output 6 Active Status			Yes		
	bit 4 – Discrete Output 5 Active Status			Yes		
	bit 3 – Discrete Output 4 Active Status			Yes		
	bit 2 – Discrete Output 3 Active Status			Yes		
	bit 1 – Discrete Output 2 Active Status			Yes		
	bit 0 – Discrete Output 1 Active Status			Yes		
3.9.2	outputControl	С	3.9 : M	YES	String	PASSED: 0x0000,0x FFFF,0x00 0000,
	Byte 1, bit 7 – Discrete Output 8 Control			Yes		
	Byte 1, bit 6 – Discrete Output 7 Control			Yes		
	Byte 1, bit 5 – Discrete Output 6 Control			Yes		
	Byte 1, bit 4 – Discrete Output 5 Control			Yes		
	Byte 1, bit 3 – Discrete Output 4 Control			Yes		
	Byte 1, bit 2 – Discrete Output 3 Control			Yes		
	Byte 1, bit 0 – Discrete Output 2 Control			Vec		
	Byte 2 bit 7 – Discrete Output 8 Active			Ves		
	Byte 2, bit 6 – Discrete Output 7 Active			Ves		
	Byte 2, bit 5 – Discrete Output 6 Active			Yes		
	Byte 2, bit 4 – Discrete Output 5 Active			Yes		
	Byte 2, bit 3 – Discrete Output 4 Active			Yes		
	Byte 2, bit 2 – Discrete Output 3 Active			Yes		
	Byte 2, bit 1 – Discrete Output 2 Active			Yes		
	Byte 2, bit 0 – Discrete Output 1 Active			Yes		
3.9.3	outputLabelIndex	Ρ	3.9 : M	YES	String	FAILED: 0x0000000 000000000 ,0x010203 040506070 8,0x08070 605040302 01,0xFFF FFFFFFF FFFFFFF FFFFFFFF FFFFFFFF FFFF
	Byte 1 – Discrete Output 1 Label Number			Yes		
	Byte 2 – Discrete Output 2 Label Number			Yes		
	Byte 3 – Discrete Output 3 Label Number			Yes		
	Byte 4 – Discrete Output 4 Label Number			Yes		
	Byte 5 – Discrete Output 5 Label Number			Yes		
	Byte 6 – Discrete Output 6 Label Number			Yes		
	Byte / – Discrete Output / Label Number			Yes		
2 10				res		
3.10	COTV Zone Objects					

CCTV Extended Functions CONFORMANCE GROUP						
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
3.10.1	zoneMaximum	S	3.10 : M	YES	0-255	16
3.10.2	zoneTable		3.10 : M	YES		
	zoneEntry		3.10 : M	YES		
3.10.2.1	zoneIndex	S	3.10 : M	YES	0-255	
3.10.2.2	zoneLabel	Ρ	3.10 : M	YES	0-255	PASSED: 0,8,255, FAILED: - 1,256,
3.10.2.3	zonePanLeftLimit	Ρ	3.10 : M	YES	1-35999 65535	PASSED: 0,35999,65 536, FAILED: - 1,35600,65 535,
3.10.2.4	zonePanRightLimit	Ρ	3.10 : M	YES	1-35999 65535	PASSED: 0,35999,36 000,65535, FAILED: - 1.65536.
3.10.2.5	zoneTiltUpLimit	Ρ	3.10 : M	YES	1-35999 65535	PASSED: - 1,36000,65 536, FAILED: 0,35999,65 535,
3.10.2.6	zoneTiltDownLimit	Ρ	3.10 : M	YES	1-35999 65535	PASSED: - 1,36000,65 536, FAILED: 0,35999,65 535,
3.10.2.7	zoneVideoControl (Added in Amend v1.08)	С	3.10 : M	YES	String	FAILED: 0x00,0x80, 0x7F,0x00 00,
	bit 7 – Video Signal Output Control			Yes		
	bit 6 – Reserved					
	bit 5 – Reserved					
	bit 4 – Reserved					
	bit 3 – Reserved					
	bit 2 – Reserved					
	bit 1 – Reserved					
	Dit U – Reserved					
3.10.3	v1.08)	S	3.10 : M	YES	String	0xe0
	bit 7 – Zones Availability			Yes		
	bit 6 – Zone Labels Availability			Yes		
	bit 5 – Video Signal Control Availability			Yes		
	bit 4 – Reserved					
	bit 3 – Reserved					
	bit 2 – Reserved					
	bit 1 – Reserved					
	bit 0 – Reserved					

B.5 CCTV MOTION CONTROL CONFORMANCE GROUP

The CCTV Motion Control Conformance Group shall consist of the following objects:

CCTV Motion Control CONFORMANCE GROUP

NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
3.4 and 3.5	CCTV Motion Control Conformance Group		0	YES		
3.4	CCTV Preset Objects					
3.4.1	presetGotoPosition	С	3.4 : M	YES	1-255	PASSED: 1,10, FAILED: - 1,0,256,
3.4.2	presetStorePosition	Р	3.4 : M	YES	1-255	PASSED: 1,10, FAILED: - 1,0,256,
3.4.3	presetPositionQuery (Added in Amend v1.08)	S	3.4 : M	YES	0-255	
3.5	CCTV Positioning Objects					
3.5.1	positionPan	С	3.5 : M	YES	String	PASSED: 0x0000000 000,0x00 00000,0x0 0000001,0 x00008C9 F,0x033F0 000,0x037 F0000,0x0 3BF0000,0 x03FF000 0,0x01013 A98,0x017 F3A98,0x0 1B13A98,0 x01FF3A9 8,0x02013 A98,0x027 F3A98,0x0 2B13A98,0 x02FF3A9 8,0x04000 000,
3.5.2	positionTilt	С	3.5 : M	YES	String	PASSED: 0x0000000 0000,0x000 00000,0x0 0000001,0 x00008C9 F,0x033F0 000,0x037 F0000,0x0 3BF0000,0 x03FF000 0,0x01013 A98,0x017 F3A98,0x0 1B13A98,0 x01FF3A9 8,0x02013 A98,0x027 F3A98,0x0 2B13A98,0 x02FF3A9 8,0x04000 000,
3.5.3	positionZoomLens	С	3.5 : M	YES	String	PASSED: 0x0000000 000,0x000 00000,0x0 0000001,0 x00008C9 F,0x033F0

	CCTV Motion Control CO	NFORMANC	E GROUP	-		
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
						000,0x037 F0000,0x0 3BF0000,0 x03FF000 0,0x01013 A98,0x017 F3A98,0x0 1B13A98,0 x01FF3A9 8,0x02013 A98,0x027 F3A98,0x0 2B13A98,0 x02FF3A9 8,0x04000 000,
3.5.4	positionFocusLens	С	3.5 : M	YES	String	PASSED: 0x000000 000,0x00 000001,0 x00008C9 F,0x033F0 000,0x037 F0000,0x0 3BF0000,0 x03FF000 0,0x04000 0,0x04000 0,0x04000 000, FAILED: 0x01013A 98,0x017F 3A98,0x01 B13A98,0x 01FF3A98, 0x02013A 98,0x027F 3A98,0x02 B13A98,0x 02FF3A98,
3.5.5	positionIrisLens	С	3.5 : M	YES	String	3A98,0x02 B13A98,0x 02FF3A98, PASSED: 0x000000 000,0x00 00000,0x0 0000001,0 x00008C9 F,0x033F0 000,0x037 F0000,0x 3BF0000,0 x03FF000 0,0x04000 0,0x037F0 0,0x04000 0,0x04000 0,0x04000 0,0x037F0 0,0x04000 0,0x04000 0,0x04000 0,0x037F0 0,0x04000 0,0x047F 3A98,0x017F 3A98,0x01 B13A98,0x02 F13A98,0x02 B13A98,0x02 F13A98,0x0

	CCTV Motion Control CONFORMANCE GROUP					
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
3.5.6	positionQueryPan (Added in Amend v1.08)	S	3.5 : M	YES	1-35999 65535	
3.5.7	positionQueryTilt (Added in Amend v1.08)	S	3.5 : M	YES	1-35999 65535	
3.5.8	positionQueryZoom (Added in Amend v1.08)	S	3.5 : M	YES	1-65535	
3.5.9	positionQueryFocus (Added in Amend v1.08)	S	3.5 : O	YES	1-65535	
3.5.10	positionQueryIris (Added in Amend v1.08)	S	3.5 : O	YES	1-65535	

B.6 CCTV ON-SCREEN MENU CONTROL CONFORMANCE GROUP

The CCTV On-Screen Menu Control Conformance Group shall consist of the following objects:

CCTV On-Screen Menu Control CONFORMANCE GROUP						
NTCIP 1205 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
3.12	CCTV On-Screen Menu Control Conformance Group		0	YES		
3.12	CCTV On-Screen Camera Menu Objects					
3.12.1	menuActivate	Ρ	3.12 : M	YES	0-255	PASSED: 0,1,254,25 5, FAILED: -1,256,
3.12.2	menuControl	С	3.12 : M	YES	1-255	
	pageDown(1)			Yes / No		
	pageUp(2)			Yes / No		
	cursorUp(3)			Yes / No		
	cursorDown(4)			Yes / No		
	cursorRight(5)			Yes / No		
	incrementValue(7)			Yes / No		
	decrementValue(8)			Yes / No		
	enterValue(9)			Yes / No		
	noMenu(255)			Yes / No		

B.7 GLOBAL CONFIGURATION CONFORMANCE GROUP

Global Configuration CONFORMANCE GROUP						
NTCIP 1201 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
2.2	Global Config Objects		М	Yes		
2.2.1	globalSetIDParmeter	S	2.2 : O	Yes / No	0-65535	
2.2.2	globalMaxModules	S	2.2 : M	Yes	0-255	
2.2.3	globalModuleTable		2.2 : M	Yes		
	moduleTableEntry		2.2 : M	Yes		
2.2.3.1	moduleNumber	S	2.2 : M	Yes	1-255	
2.2.3.2	moduleDeviceNode	S	2.2 : M	Yes	OID	
2.2.3.3	moduleMake	S	2.2 : M	Yes	String	
2.2.3.4	moduleModel	S	2.2 : M	Yes	String	
2.2.3.5	moduleVersion	S	2.2 : M	Yes	String	
2.2.3.6	moduleType	S	2.2 : M	Yes	1-3	
	other(1)			Yes / No		
	hardware(2)			Yes / No		
	software(3)			Yes / No		

The Global Configuration Conformance Group shall consist of the following objects:

B.8 NTCIP SECURITY CONFORMANCE GROUP

The NTCIP Security Conformance Group shall consist of the following object
--

Security CONFORMANCE GROUP						
NTCIP 1201 Amend 1 Clause	Object Name	Object Type	Object Status	Object Support	Allowed Values	Supported Values
A.10	Security Conformance Group		M	Yes		
A.10.1	adminCommunityName	С	A.10 : M	Yes	String	
A.10.2	maxCommunityNames	С	A.10 : M	Yes	1255	
A.10.3	communityNameTable		A.10 : M	Yes		
	communityNameTableEntry		A.10 : M	Yes		
A.10.3.1	communityNameIndex	S	A.10 : M	Yes	1255	
A.10.3.2	communityNameUser	S	A.10 : M	Yes	String	
A.10.3.3	communityNameAccessMask	S	A.10 : M	Yes	Gauge	

§

REFERENCES FOR APPENDIX D

 NTCIP 1205 – Object Definitions for Closed Circuit Television (CCTV) Camera Control, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=1205</u>. Accessed July 25, 2005.

APPENDIX E: COMMUNICATIONS AND MISCELLANEOUS TEST PROCEDURES

INTRODUCTION

This appendix describes several sets of test procedures that may be applicable to NTCIP conformant field devices. The test procedures include procedures for checking the Simple Network Management Protocol (SNMP), the Transportation Transport Profile, the Point to Multi-Point Protocol with RS232 Profile, the Simple Transportation Management Protocol (STMP), and serial communication's data rates. The original basis of the test cases was the requirements in NTCIP 2001 – Class B Profile (*1*). The NTCIP standards group has subsequently rescinded the original standard and replaced it with several others. The documentation and summaries come from the NTCIP Laboratory Testing for Actuated Signal Controllers summary report (*2*). Test scripts implementing the test procedures are available at www.itstestlab.org.

SNMP TEST CASES

Table E-1 summarizes a set of test cases for checking implementations supporting the SNMP. Except for the SNMP Conformance Group test case, the procedures check the functional aspects of SNMP and are not object related. The test cases are generic but require support of an object with a specific syntax and constraint that may not be available in all field devices.

SNMP Test Cases				
ID	Title	Description		
General				
TC001	Setup	General setup for testing SNMP		
TC002	MIB Walk	This test ensures that GETNEXT requests are properly functioning.		
TC003	Get and Set an Object	This test ensures that GET and SET operations return a valid value and do not produce errors.		
TC004	Get and Set Multiple Objects	This test ensures that multiple objects can be set correctly with SET operation.		

	SNMP Test Cases			
ID	Title	Description		
Error Responses				
TC005	Error In Get of Multiple Objects (tooBig)	This test ensures that GET function returns a tooBig error when a response protocol data unit (PDU) exceeds snmp-maxPacketSize.		
TC006	Error in Set (badValue)	This test ensures that a bad (Invalid) value is not set using SET function.		
TC007	Error in Set (readOnly)	This test ensures that SET function cannot change value of a 'readOnly' object.		
TC008	Error in Set (noSuchName)	This test ensures that a value cannot be set for a non- existing object.		
TC009	Error in Setting Multiple Objects (badValue)	This test ensures that while setting multiple objects, if one object is set to a bad value then other valid values are not changed for the other objects.		
Community Name				
TC010	Invalid Community Name	This test ensures that if community name is invalid then no object is returned with SET function.		
	Statistic	s Conformance Group		
TC011	SNMP Conformance Group	This procedure tests whether 2 objects in the SNMP (Statistics) Conformance Group are instantiated.		
		Encoding		
TC012	INTEGER Encoding	Check whether an object with "SYNTAX INTEGER" can accept an instance value of 0 when the length byte is set to 1, 2, 3, 4, and 5 bytes using the "short definite form" and when the "long definite form" is used. Also checks setting values of >2147483647.		
TC013	INTEGER (0255) Encoding	Check whether an object with "SYNTAX INTEGER (0255)" can accept an instance value of 0 when the length byte is set to 1 and 2 bytes using the "short definite form" and when the "long definite form" is used.		

	SNMP Test Cases					
ID	Title	Description				
TC014	INTEGER (065535) Encoding	Check whether an object with SYNTAX INTEGER (065535) can accept an instance value of 0 when the length byte is set to 3 bytes using both the "short definite form" and the "long definite form."				
TC015	INTEGER (04294967295) Encoding	Check whether an object with SYNTAX INTEGER (04294967295) [INTEGER Tag 02] can accept an instance value of 0 when the length byte is set to 3 bytes using both the short definite form and the long definite form.				
TC016	INTEGER Wrong Tag Encoding	Check whether an object as SYNTAX INTEGER (0255) [INTEGER Tag 02] would can accept an instance value with an incorrect Tag (OCTET STRING, OBJECT IDENTIFIER, SEQUENCE, IpAddress, Counter, Gauge, TimeTicks, Opaque).				
TC017	OCTET STRING Encoding	Check whether an object with OCTET STRING [Tag 0x04] can accept an instance value of NULL [Tag 0x05], a 4 character string using the short definite length form, an 127 character string using the short definite length form, and a 2 each - 4 character strings using the long definite length form.				
TC018	OCTET STRING Constrained Encoding	Check whether an object with SYNTAX OCTET STRING [Tag 0x04] and a size constraint is processed correctly.				
TC019	Object Identifier and Null Encoding	Check whether an object with SYNTAX OBJECT IDENTIFIER [Tag 0x06] would accept as an instance value of NULL [Tag 0x05] and a valid length.				
TC021	Counter Encoding	Check whether an object with SYNTAX Counter [Tag 0x41] is processed correctly.				
TC022	Counter (0255) Encoding	Check whether an object with SYNTAX Counter (0255) [Tag 0x41] is processed correctly.				
TC023	Counter (065535) Encoding	Check whether an object with SYNTAX Counter (065535) [Tag 0x41] is processed correctly.				
TC024	Gauge Encoding	Check whether an object with SYNTAX Gauge [Tag 0x42] is processed correctly.				
TC025	Gauge (0255) Encoding	Check whether an object with SYNTAX Gauge (0255) [Tag 0x42] is processed correctly.				
TC026	Gauge (065535) Encoding	Check whether an object with SYNTAX Gauge (065535) [Tag 0x42] is processed correctly.				

SNMP Test Cases			
ID	Title	Description	
TC027	TimeTicks Encoding	Check whether an object with SYNTAX TimeTicks [Tag 0x43] is processed correctly.	
	Opaque Encoding		
TC028	Opaque Encoding – Setup	This procedure performs a general setup prior to executing any of the specific Opaque Encoding Test Procedures.	
TC029	Opaque Encoding – INTEGER	Check whether an object with SYNTAX Opaque [Tag 0x44] that encodes an instance of INTEGER is processed correctly.	
TC030	Opaque Encoding - INTEGER (0255)	Check whether an object with SYNTAX Opaque [Tag 0x44] that encodes an instance of INTEGER (0255) is processed correctly.	
TC031	Opaque Encoding - INTEGER (065535)	Check whether an object with SYNTAX Opaque [Tag 0x44] that encodes an instance of INTEGER (065535) is processed correctly.	
TC032	Opaque Encoding - INTEGER (04294967295)	Check whether an object with SYNTAX Opaque [Tag 0x44] that encodes an instance of INTEGER (04294967295) is processed correctly.	
TC033	Opaque Encoding - INTEGER (Constrained)	Check whether an object with SYNTAX Opaque [Tag 0x44] that encodes an instance of INTEGER (Constrained) is processed correctly.	
TC034	Opaque Encoding - OCTET STRING	Check whether an object with SYNTAX Opaque [Tag 0x44] that encodes an instance of OCTET STRING is processed correctly.	
TC035	Opaque Encoding - OBJECT IDENTIFIER	Check whether an object with SYNTAX Opaque [Tag 0x44] that encodes an instance of OBJECT IDENTIFIER is processed correctly.	
TC036	Opaque Encoding - IpAddress	Check whether an object with SYNTAX Opaque [Tag 0x44] that encodes an instance of IpAddress is processed correctly.	
TC037	Opaque Encoding - Counter	Check whether an object with SYNTAX Opaque [Tag 0x44] that encodes an instance of Counter is processed correctly.	
TC038	Opaque Encoding - Gauge	Check whether an object with SYNTAX Opaque [Tag 0x44] that encodes an instance of Gauge is processed correctly.	

SNMP Test Cases		
ID	Title	Description
TC039	Opaque Encoding - TimeTicks	Check whether an object with SYNTAX Opaque [Tag 0x44] that encodes an instance of TimeTicks is processed correctly.
TC040	Opaque Encoding - Tear Down	This procedure performs a general restore after executing any of the specific Opaque Encoding Test Procedures.

TRANSPORTATION TRANSPORT TEST CASES

Table E-2 summarizes several test cases for checking implementations supporting the NTCIP 2201 – Transportation Transport Profile (*3*). The first two test cases test the functionality of the protocol. The Net to Media Support test case relates to an object conformance group but the group is mandatory only if an IP Address scheme is used.

Table E-2. Transportation Transport Test Case Summary.

Transportation Transport Profile		
ID	Title	Description
NULL		
TC001	Unknown IPI	This test checks whether a device under test (DUT) accepts an upper layer Protocol Data Unit with an invalid Initial Protocol Identifier.
TC002	Max Protocol Data Unit	This procedure checks whether a DUT supports the required Protocol Data Unit size.
TC003	Net to Media Support	This procedure checks for support of the ipNetToMedia conformance group.

POINT TO MULTI-POINT WITH RS232 TEST CASES

Table E-3 summarizes a set of test cases for checking implementations supporting the NTCIP 2101 – Point to Multi-Point Protocol Using RS-232 Subnetwork Profile (4). Except for the RS232 Conformance Group and the LapB Conformance Group test cases, the procedures check the functional aspects of PMPP and are not object related.

PMPP with RS-232			
ID	Title	Description	
Short Address			
TC001	Setup	General setup for PMPP Procedures	
TC002	Short Address - Positive Test 1	This procedure checks whether a DUT responds to a valid short address.	
TC003	Short Address - Negative Test 1	This procedure checks whether a DUT responds to another DUT's address.	
TC004	Short Address - Positive Test 2	This procedure checks whether a DUT responds to another valid short address.	
TC005	Short Address - Negative Test 2	This procedure checks whether a DUT responds to another DUT's short address.	
TC006	Restore Default Address	This test ensures that the default address is restored properly.	
	Long Address		
TC007	Long Single Address - Positive Test	This procedure checks whether a DUT responds to a valid long address.	
TC008	Large Single Address - Negative Test #1	This procedure checks whether a DUT responds to another DUT's long address.	
TC009	Large Single Address - Negative Test #2	This procedure checks whether a DUT responds to an invalid long address.	
TC010	Restore Default Address	This test ensures that the default address is restored properly.	
Broadcast and Polling			
TC011	Broadcast Message	This procedure checks whether a DUT responds to a broadcast message or not.	

Table E-3. PMPP with RS-232 Test Case Summary.

PMPP with RS-232		
ID	Title	Description
TC012	Poll Message	This test ensures that previous broadcast message's
		(UP) is sent to the DUT.
		Group Address
TC013	Setting Group Address	This test ensures that the group address is set properly.
TC014	Group Address - Positive	This procedure checks whether a DUT does not
	lest	respond to a GET with a valid group address but
		upon a UP.
TC015	Group Address - Negative	This procedure checks whether a DUT responds to a
	Test	GET with an invalid group address.
TC016	Large Group Address -	This procedure checks whether a DUT responds to a
TC017	Positive Test	GET with a valid large group address.
10017	Large Group Address - Negative Test	GET with an invalid large group address
_		
		Polling
TC018	Request Without Poll Bit	This procedure checks whether a DUT responds to
		GET request to valid single address but without the
TC 010		Poll Bit set.
1C019	Poll	This procedure checks whether a DUT responds with the huffored response from the Dequest Without Poll
		Bit procedure when a UP is sent.
	L	
		Control Byte
TC020	Changed Control Byte	This procedure checks whether a DUT responds to a
		PDU with an invalid Control Byte.
Initial Protocol Identifier (IPI)		
TC021	Unknown IPI	This procedure checks whether a DUT responds to a
		PDU with an invalid IPI.

Table E-3. PMPP with RS-232 Test Case Summary (continued).

Field Check Sum

PMPP with RS-232		
ID	Title	Description
TC022	Invalid Cyclic Redundancy Check (CRC)	This procedure checks whether a DUT responds to a PDU with an invalid CRC on good data.
TC023	Changed Data	This procedure checks whether a DUT responds to a PDU with a valid CRC on bad data.
RS232 and HDLC Conformance Groups		
TC024	RS232 Conformance Group	This procedure checks whether a DUT responds to one of the objects in the RS232 Conformance Group.
TC025	HDLC Conformance Group	This procedure checks whether a DUT responds to one of the objects in the HDLC Conformance Group.
Frame Size and Buffering		
TC026	Frame Size	This procedure checks whether a DUT can accept and send a PDU that is the largest that must be supported.
TC027	Byte Stuffing	This procedure checks whether the DUT can support a PDU where 5% of the octets are byte-stuffed.

Table E-3. PMPP with RS-232 Test Case Summary (continued).

STMP TEST CASES

Table E-4 summarizes a set of test cases for checking implementations supporting the Simple Transportation Management Protocol (STMP), whose definition appears in NTCIP 1103 – Transportation Management Protocols and whose profile requirements appear in NTCIP 2301 – Simple Transportation Management Framework Application Profile (5,6). The procedure defines a set of dynamic object messages that encode the various data types and then checks that the encoded data type values are valid. The documentation and test scripts for these procedures are only partially complete. However, they do provide a starting point and approach to testing the functional aspects of STMP protocol.
	STMP Test Cases					
ID	Title	Description				
TC001	General Setup	This procedure clears out or invalidates any previous dynamic object definitions.				
TC002	Set Up Dynamic Objects 1 To 11 to a Single Variable	This procedure defines a set of dynamic messages consisting of a single variable.				
TC003	Define Compare Values	This procedure defines a set of compare values objects for use in the single variable test.				
TC004	Compare STMP results	This procedure compares the values of the single variable messages test returned using STMP with those returned using SNMP.				
TC005	Set Up Dynamic 12 to 11 Variables	This procedure defines a single dynamic message consisting of 11 variables.				
TC006	Compare STMP Object 12 Results	This procedure compares the values of the 11 variable dynamic messages returned using STMP with those returned using SNMP.				
TC007	General Clean Up	This procedure invalidates the dynamic objects definitions.				

Table E-4. STMP Test Case Summary.

RESPONSE TIME TEST CASE

Table E-5 summarizes a test case for checking support of the various data rates as defined in NTCIP 2101 – Point to Multi-Point Protocol Using RS-232 Subnetwork Profile and in NTCIP 2102 – Point to Multi-Point Protocol Using FSK Modem Subnetwork Profile (4,7). By repeating a message exchange multiple times, one can make an evaluation of the response time or length of time that an implementation takes to process a message.

	PMPP Data Rates and Response Times				
ID	Title	Description			
TC001	Response Time	The purpose of this test procedure is to evaluate response time and test for support of various data rates defined in NTCIP 2101 and 2103. NTCIP 2101 requires 1200 bps and lists 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, and others as optional. NTCIP 2102 is 1200 bps by definition. NTCIP 2103 requires support for 2400, 4800, 9600, 19.2K bps and states that higher data rates are optional.			

Table E-5. Response Time Test Case Summary.

REFERENCES FOR APPENDIX E

- *1*. NTCIP 2001 Class B Profile, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=2001</u>. Accessed August 16, 2006.
- 2. NTCIP Laboratory Testing for Actuated Signal Controllers, Summary Report for ASSHTO Project 475070. Published by Texas Transportation Institute. <u>http://tti.tamu.edu/documents/TTI-2006-1.pdf</u>. Accessed June 7, 2006.
- 3. NTCIP 2201 Transportation Transport Profile, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=</u> <u>no&standard=2201</u>. Accessed July 25, 2005.
- 4. NTCIP 2101– Point to Multi-Point Protocol Using RS-232 Subnetwork Profile, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=2101</u>. Accessed June 27, 2006.
- 5. NTCIP 1103 Transportation Management Protocols, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> d=1103. Accessed June 27, 2006.
- 6. NTCIP 2301– Simple Transportation Management Framework Application Profile, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=2301</u>. Accessed June 27, 2006.
- 7. NTCIP 2102– Point to Multi-Point Protocol using FSK Modem Subnetwork Profile, A Joint Publication of AASHTO, ITE, and NEMA.
 <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=2102</u>. Accessed June 27, 2006.

APPENDIX F: TRAFFIC SIGNAL CONTROLLER TEST DOCUMENTATION

INTRODUCTION

Since the task of developing NTCIP test procedures for one of the ITS field devices can be a significant project in and of itself, the researcher found that following some of the recommendations in the Institute of Electrical and Electronics Engineers (IEEE) Std. 829 - IEEE Standard for Software Test Documentation are useful (1). Prior to actually writing procedures, the IEEE standard suggests the development of an overall plan, one or more test design specifications, and test case specifications. The overall plan conveys the scope, approach, resources, and schedule of testing activities. Its primary purpose is to present a high-level view of the project to inform all interested parties. The test design specifications provide a more detailed view of the testing project. A test engineer's supervisor and any group such as a project monitoring committee uses the test design specifications to make sure that a test engineer understands the projects and is addressing what is needed. The design specification serves as part of the validation step in the project development. Test case specifications then outline individual test cases that verify specific features and functions of an implementation undergoing test. The test case specifications provide additional oversight but primarily help a test engineer organize and plan the specifics of each test case before committing to code or formal definition. These types of documents address the planning aspects of a testing project.

Full development of these documents is beyond the scope of this TxDOT research project. In the case of the CCTV test procedures, the researchers capitalized on test procedures already in the public domain, and upfront planning did not appear to be essential. For the traffic signal controllers, however, the researcher felt that it would be helpful to have a test design specification and test case specification that organizes and outlines the approach to testing the requirements that would apply to TxDOT department material specification DMS-11170, Fully Actuated, Solid-State Traffic Signal Controller Assembly (2). A test design specification and test case specification from a previous project were somewhat appropriate and can serve to convey the makeup of such documents (*3*).

203

A number of state departments of transportation are adopting International Organization for Standardization (ISO) 9000 standards in order to improve quality (4). The following test design specification and test case specification are two types of document examples that would satisfy most of the ISO 9000 requirements. The following embedded specifications, initially prepared under another research project, have been modified to put them in the context of this research.

Test Design Specification

NTCIP Conformant Traffic Signal Controller

TDS-TSC v1.02

August 31, 2006

REVISION HISTORY

Revision Date	Version Number	Description of Change
03/08/06	v1.01	Initial draft by R. De Roche
08/31/06	v1.02	Revised for inclusion in TxDOT project 0-5003

Test Design Specification

1.0 INTRODUCTION

This test design specification outlines the requirements for testing NTCIP compliant traffic signal controllers. This specification identifies the features and/or general functions to be tested. This specification also details the test approach, proposes a rationale for the definitions of the test cases, and establishes pass/fail criteria.

This test design specification defines the elements and approach to show compliance to the NTCIP related requirements of TxDOT DMS-11170, Fully Actuated, Solid-State Traffic Signal Controller Assembly (2). This document identifies the elements to test and those that will not. The test design specification also identifies the test cases and subsequent test procedures. A test case specification documents the actual values used in the testing process. The test case specification also identifies constraints on the test procedures. Test cases are separate from test designs so that test cases can apply to one or more design specifications and have application in other test plans. A test procedure identifies the specific steps involved in executing a test case. By their nature, test procedures go into detail about a systematic process. Test procedures are in a separate document so that they do not burden the other documents with extraneous detail.

To meet the first objective, tests will be designed to evaluate a DUT's conformance to the appropriate NTCIP Standards. The current versions of the standards convey requirements in the form of a Profile or Protocol Requirement List (PRL). The testing results will be primarily conveyed to participants in the Test Bed Project by recording the results on the PRL and, thus, turning it into a Profile or Protocol Implementation Conformance Specification.

To meet the second objective, tests will be designed to evaluate a DUT's compliance to additional requirements that might be imposed when a device is used in a system. These additional requirements are defined within this document and will be summarized in the form of a Device Requirements List (DRL) as defined in Annex A. The testing results will be primarily conveyed to participants in the Test Bed Project by recording the results on the DRL and, thus, turning it into a Device Implementation Conformance Specification.

This document's organization and content uses IEEE Std. 829 – IEEE Standard for Software Test Documentation as a guide (1). The purpose of the IEEE standard is to define a basic set of testing documentation that provides a common point of reference for discussion and understanding for all parties involved in the testing process. The design of the contents serves as a completeness checklist.

1.1 TEST DESIGN SPECIFICATION IDENTIFICATION

The Test Design Specification Identifier is TDS-TSC.

1.2 FEATURES TO BE TESTED

The TxDOT specification DMS-11170, Fully Actuated, Solid-State Traffic Signal Controller Assembly defines the requirements for traffic signal controllers and other components in a signal cabinet. The organization of requirements in that specification that relate specifically to controller units is:

- Hardware Design Requirements NEMA Controller
- Time Clock
- Clock-Calendar Programming
 - Structure and Interrelationship of Programs
- Programming Requirements
 - Phase Operation

- Pedestrian Timing
- o Coordination
- Time Base Coordination
- Diamond Operation
 - o Program Requirements
 - Four-Phase Operation
 - Concurrent Timing Requirements
 - Diamond Detector Operation
 - o Three-Phase Operation
- Coordination-Control Hierarchy
- Preemption
- Closed-Loop Operation and Monitoring Software
- NTCIP Compliance

This version of the test design specification focuses specifically on:

- NTCIP Compliance
- Four-Phase Operation

NOTE – Other than Diamond Four-Phase Operation, this test design specification does not cover test cases for the functional requirements of a signal controller.

The basic features and functions associated with the NTCIP Compliance requirements of DMS-11170 are:

- Object support for mandatory objects defined in:
 - o Phase Conformance Group
 - o Detector Conformance Group
 - Volume Occupancy Report Conformance Group
 - Unit Conformance Group
 - Special Function Conformance Group
 - o Coordination Conformance Group
 - Time Base Conformance Group
 - Time Management
 - Time Base Event Schedule
 - o Preempt Conformance Group
 - Ring Conformance Group
 - Channel Conformance Group
 - o Overlap Conformance Group
 - TS 2 Port 1 Conformance Group
 - Configuration Conformance Group
 - o Database Management Conformance Group
 - Report Conformance Group
 - PMPP Conformance Group
 - STMF Conformance Group
 - SNMP Conformance Group
 - STMP Conformance Group
 - Security Conformance Group
 - HDLC Group Address Conformance Group
- Object support for miscellaneous optional objects defined in
 - o NTCIP 1201-GLO
 - o NTCIP 1202-ASC
- Support Standardized Ranges
- Specific Values Defined in Minimum Project Requirements
- Null Protocol
- PMPP Protocol using an RS-232 Physical Interface
- PMPP Protocol using an FSK Modem Interface

Additional features and functions not defined within DMS-11170 but considered relevant to controller operation:

- Block Object Conformance Group
- Systems Conformance Group
- SNMP Statistics Conformance Group
- RS232 Conformance Group
- HDLC Conformance Group

1.3 FEATURES NOT TO BE TESTED

The following features and functions do not appear as requirements in DMS-11170 and appear in NTCIP 1202 as optional. They are not essential to controller operation and, therefore, will not be included:

- Auxiliary I/O Group
- SFMP Group
- Logical Name Group
- Trap Management Group
- Interfaces Group
- IP Group
- ICMP Group
- TCP Group
- UDP Group
- Ethernet Group

1.4 APPROACH REFINEMENTS

1.4.1 Conformance Group, Optional Objects, Standardized Ranges, and Project Requirements

For the following three test cases, one or more MIBs will consolidate all the required objects and eliminate non-required objects.

1.4.1.1 Conformance Group and Optional Object Support

For testing support of the conformance groups and optional objects, a test case will perform a "MIB Walk." This MIB walk, however, will be somewhat different from what others consider to be a MIB walk. The traditional MIB walk uses SNMP Get Next operations to step through the objects that an implementation supports. The Get Next operation returns the object identifier of the next object in logical sequence. Comparing this object identifier value to what appears in a MIB can indicate what objects are present or missing. The Get Next operation also "discovers" any additional objects. While this is a good method for determining what an implementation supports and checking the Get Next operation, it does not use a MIB as the reference point or use Get operations. From the perspective of an agency purchasing equipment, the primary concern is whether equipment implements their requirements. Knowing about additional objects can be useful but the required ones are the concern. The normal operational method of retrieving object values is through the Get operation.

Given these reasons, the basis of the MIB Walk will be Get operations. Applying potential instance value extensions to the list of object names in the MIB will generate a hierarchical set of object identifiers to use in the Get operation. By stepping through each object identifier, a test procedure can determine an implementation's object support.

NOTE – This test case will only check for the presence of objects and will not check the functionality expressed by them.

1.4.1.2 Supported Values of Instantiated Objects

For testing the supported values, a test case will rely on an external list of sample test values to use. Test cases that perform a 100% check of all the possible values are relatively easy to define. However, when

one considers that a controller contains tens of thousands of objects and the typical value range of these objects is 0 to 255, the time it takes to perform a test becomes an issue. Rather than use this approach, the test case will use a sampling technique. For every read-write object that corresponds to a parameter or control, the external list will define the objects to test. The objects will also have both positive and negative test values to use on the objects.

Each object that has only a single instance will appear in the external list. At least one instance of an object that appears multiple times in a table structure should also appear in the external list. Along with the actual test value to use, there is an indication of whether the value a positive or negative test value.

NOTE – The supported values test case will only check whether an object can be set to specific values. It does not check that the written values are actually used or the controller performs the functionality expressed by a value.

1.4.1.3 Specific Values Defined in Minimum Project Requirements

The Object Range Value for Actuated Signal Controllers table in DMS-11170 refers to two types of objects: "max" objects or constants and status objects that indicate values that correspond to states of the controller. For max objects, a test case will use Get operations to retrieve the value in the implementation and then compare the value against the required value in DMS-11170. To make this test case reusable for testing equipment other than signal controllers, it will use an external list of object identifiers to determine which objects values to retrieve. Associated with each object will be the minimum project requirement value for comparison.

NOTE – The max object test case will only check for the value of the max objects. The test case will not check the functionality expressed by them. In most cases, a max object defines a number of instances. For example, the maxPhases object defines the number of phases that a controller supports. Checking the value of maxPhases does not determine whether a controller implements that number of phases.

Status objects are more difficult to validate. Status objects take on a specific value only when the controller is in a specific state. For each status object, a test case will create the conditions that produce each state. The test case will then either verify the correct state through independent means or assume some time for the controller to reach the state. The test case will then Get the value of the status object and compare to the appropriate value.

NOTE – The status object test case will test the functionality expressed by the status object either directly or indirectly.

1.4.2 SNMP Protocol

The conformance testing of the SNMP protocol will involve a number of test cases. The test cases will have the following organization:

- General
- Error Responses
- Community Name
- SNMP Statistics Conformance Group
- BER Encoding
- Opaque Encoding

1.4.3 STMP Protocol

The conformance testing of the STMP protocol will consist of defining twelve dynamic messages to retrieve object definitions that require different Octet Encoding Rules (OER). The intent is to check the encoding of the following SYNTAX types:

- INTEGER [unconstrained]
- INTEGER (0..255)
- INTEGER (0..65535)

- INTEGER (0..4394967295)
- INTEGER (-43200..43200)
- OCTET STRING [unconstrained]
- OBJECT IDENTIFIER
- IpAddress
- Counter
- Gauge
- TimeTicks

The first eleven messages will consist of a single variable corresponding to one of the syntaxes. The twelfth message will consist of eleven objects corresponding to all of the syntax types.

NOTE – The conformance tests will not address the OER encoding of the syntax Opaque.

1.4.4 Null Protocol

The conformance testing of the Null Protocol (now referred to as the Transportation Transport Profile) will involve three test cases. These will consist of:

- Unknown IPI
- Max Protocol Data Unit
- Net to Media Support

1.4.5 PMPP Protocol Using an RS-232 Physical Interface

The conformance testing of the Point-to-Point Protocol using an RS-232 will be broken down into ten areas. These areas consist of:

- Short Address
- Long Address
- Broadcast and Polling
- Group Address
- Polling
- Control Byte
- Initial Protocol Identifier
- Field Check Sum
- RS232 and HDLC Conformance Groups
- Frame Size and Buffering

1.4.6 PMPP Protocol Using an FSK Modem Interface

The conformance testing related to the use of an FSK modem will consist of using a line impairment device to simulate worst-case conditions of the transmission line and then checking for errors. The name of the test case will be:

• Bell 202T Modem Characteristics

1.4.7 Diamond Four-Phase Operation

The test cases for diamond four-phase operation will consist of two types. The first type will address sequencing, and the second type will address detector operations.

The sequencing test cases will cycle the controller through the possible sequence patterns. In that fourphase sequence there are 6 primary movements or states: 2+5, 3+5, 4+5, 1+6, 1+7, and 1+8. This equates to 15 permutations of one state transitioning to another. The first part of the sequence test will be to put the controller in one state and by means of detector calls, get it to change to another state and then back again. Table 1 lists the initial state, detector calls that are active, and the resulting sequence.

State	Calls	Sequence	Remarks
2+5	2, 3	3+5, 2+5 →	
2+5	2, 4	4+5, 2+5 →	
2+5	2, 6	5+9, 1+6, 1+13, 2+5 →	
2+5	2, 7	5+9, 1+7, 2+15, 2+5 →	
2+5	2, 8	5+9, 1+8, 2+16, 2+5 →	
3+5	3, 4	4+5, 3+5 →	
3+5	3, 6	6+11, 1+6, 1+13, 3+5 →	
3+5	3, 7	5+9, 1+7, 1+13, 3+5 →	
3+5	3, 8	5+9, 1+8, 1+13, 3+5 →	
4+5	4, 6	6+12, 1+6, 1+13, 4+5 →	
4+5	4, 7	5+9, 1+7, 1+13, 4+5 →	
4+5	4, 8	5+9, 1+8, 1+13, 4+5 →	
1+6	6, 7	1+7, 1+6 →	
1+6	6, 8	1+8, 1+6 →	
1+7	7, 8	1+8, 1+7 →	

Table 1. Sequence Table with Return to Initial State

1.4.8 Diamond Four-Phase Detector Operations

The detector operations test cases will address each detector in turn. Since most of the detectors operations involve switching, calling, and extending, test cases will create conditions that represent the states of the controller, activate the detector input, and then monitor the status of the traffic signal controller for the proper response.

1.4.9 Additional Test Cases

The following test cases from the Test Bed Project will be applied to the controller, with time permitting:

- Retrieve Log Data
- Timebase Schedule of Event
- Database Upload/Download
- System Performance Testing

1.5 TEST IDENTIFICATION

The test case specifications associated with this Test Design Specification is TCS-TSC.

1.6 FEATURE PASS FAIL CRITERIA

The Conformance Group, Optional Objects, Standardized Ranges, and Project Requirements pass-fail criteria depend upon individual test parameters. The procedures will use the NTCIP 1202-ASC Profile Requirements List (Annex A) to record results.

The other test cases will define any specific pass-fail criteria but in general, the device under test must perform the stated operation or return the expected results to "pass."

Test Bed Project Test Case Specifications Actuated Signal Controller

ITL-TCS-TBP-ASC v1.10

August 31, 2006

REVISION HISTORY

Revision Date	Version Number	Description of Change		
4/17/03	v1.01	Initial draft for review by C. Herrick		
5/20/03	v1.02	Incorporated various comments from C. Herrick		
6/2/03	v1.03	Various updates to most clauses		
1/26/04	v1.04	Updated title and added document organization		
4/6/04	v1.05	Changed title and ID, segmented test cases into their own		
		sections		
6/2/04	v1.06	Added filename at end and miscellaneous edits		
		Correct Identifier		
7/1/04	v1.07	Added J. Johnson's corrections		
7/9/04	v1.08	Completely revised Retrieve		
8/14/06	v1.09	Revised for TxDOT Project 0-5003		
8/31/06	v1.10	Removed copyright		

Test Case Specification

A Test Case Specification describes precisely what is to be tested. It requires identification for each test case, a description of the test items, a reference to the functions to be tested, the inputs and expected outputs, and the test case dependencies.

This organization of documents in the IEEE Std. 829 is:

- Test Plan
- Test Design Specification
- Test Case Specification
- Test Procedure Specification

1.7 TEST CASE SPECIFICATION IDENTIFIER

This Test Case Specification Identifier is:

ITL-TCS-TBP-ASC

This Test Case Specification addresses each of the major test cases as defined in the Test Design Specification (ITL-TDS-TBP-ASC). The major test cases are:

- Object Instantiation of NTCIP 1202 and 1201
- Supported Values of Instantiated Objects
- SNMP Protocol
- Null Protocol
- PMPP using RS-232
- PMPP using FSK Modem
- System Operational Scenarios
- Retrieve Log Data Optional Operational Scenario
- Timebase Schedule of Event and Database Upload/Download Optional System Operational Scenarios
- System Performance Testing

Each specific test case is covered in a separate section of this document. Additional and supporting documentation appears as Annexes.

Object Instantiated Test Case

1.8 TEST ITEMS

This test case applies to devices that implement the data elements defined within NTCIP 1202 Object Definitions for Actuated Signal Controllers. While NTCIP 1202 is, technically, an information profile that defines the functional data elements related to an Actuated Signal Controller, it also enumerates other data elements that would be instantiated in a fully conformant implementation. For example, NTC1202 references data elements for RS-232 interfaces and Ethernet interfaces. While data elements related to these elements are not mandatory, if a device supports the functionality expressed by them, then they should be supported. This test case specification only deals with the objects that are defined in the MIB. All MIB defined objects are checked for instantiation and the range of values supported.

1.8.1 Requirement Specifications

The requirements specifications are defined in NTCIP 1202 and are summarized in NTCIP 1202 v02.18 - Annex A. The following appears at the beginning of Annex A:

Conformance Groups are defined as either mandatory or optional. If a Conformance Group is mandatory, all of the objects with STATUS "mandatory" that are part of the Conformance Group shall be present for a device to claim conformance to the Conformance Group. If a Conformance Group is optional, all of the objects that are part of the Conformance Group with the STATUS "mandatory" shall be present if the device supports the Conformance Group. Objects with the STATUS "optional" may be supported.

When a table is included in a Conformance Group, all objects contained in the table are included by reference. This is because a table is defined as a SEQUENCE OF {SEQUENCE}. Thus, all objects listed in the sequence are defined as an integral part of the table. Tables are defined as either mandatory or optional. If a table is mandatory, all of the objects with STATUS "mandatory" shall be present. If a table is optional, all of the objects with the STATUS "mandatory" shall be present if the device supports the table. Objects in the table with the STATUS "optional" may be supported.

The following statements appear in Clause A.2:

Additional objects or groups may be supported without being non-compliant with ASC objects or NTCIP. Minimum and maximum ranges of objects that differ from the values of the object's SYNTAX field may be enforced by an application running on a device.

A device which enforces range limits within the bounds specified by the values of the object's SYNTAX field shall not be categorized as being non-compliant with ASC objects or NTCIP.

A device which supports a subset of objects with enumerated values shall not be categorized as being non-compliant with ASC objects or NTCIP.

The table in Clause A.2 indicates that the following conformance groups defined within NTCIP 1202 shall be supported:

- Phase Conformance Group
- Detector Conformance Group

The table in Clause A.2 indicates that the following conformance groups defined within other standards shall be supported:

- Configuration Conformance Group
- Database Management Conformance Group

- SNMP Group
- Systems Group
- Security Group

Clause A.2 also indicates that the following conformance groups defined within NTCIP 1202 may be optionally supported:

- Volume Occupancy Report Conformance Group
- Unit Conformance Group
- Special Function Conformance Group
- Coordination Conformance Group
- Time Base Conformance Group
- Preempt Conformance Group
- Ring Conformance Group
- Channel Conformance Group
- Overlap Conformance Group
- TS 2 Port 1 Conformance Group
- Block Object Conformance Group

Clause A.2 also indicates that the following conformance groups defined within other standards may be optionally supported:

- Report Conformance Group
- PMPP Group
- STMP Group
- Logical Name Group
- Trap Management Group
- RS232 Group
- HDLC Group
- Interfaces Group
- IP Group
- ICMP Group
- TCP Group
- UDP Group
- Ethernet Group

The purpose of this test case is to determine what data elements (objects definitions) are supported (instantiated) in an implementation. While NTCIP 1202 is, technically, an information profile that defines the data elements related to an Actuated Signal Controller, it also enumerates other data elements that would be instantiated in a fully conformant implementation. For example, NTC1202 references data elements for RS-232 and Ethernet Interfaces. NTCIP 1202 does not define these but does reference them. While the data elements for RS-232 and Ethernet are not mandatory, if a device supports the functionality expressed by them, then they should be supported. This test case specification not only deals with the objects that are defined in the NTCIP 1202 MIB but also those referenced in other standards.

1.8.2 Design Specifications

There are two perspectives to this test case. What objects does a DUT support and what objects defined in NTCIP 1202 are supported by a DUT.

1. In the context of SNMP, a "MIB Walk" can be used to discover what objects a DUT supports. This set of objects would include not only NTCIP 1202 defined data elements but also "any" data element, be it either NTCIP defined or not. A "Get Next" of the root Object Identifier that defines NTCIP objects can determine what objects are supported. Comparing this list of supported objects to those defined in the standard verifies whether an object is instantiated or not.

2. Given the Object Identifiers of the data elements defined in the standard and the values defined in a DUT to indicate the number of "rows" in certain tables, a list of fully indexed Object Identifiers can be created. A "Get" of each object identifier in the list can verify whether each object is instantiated.

A set of test procedures are to be developed using both of these methods. The test procedures for using the "Get Next" approach shall be referred to as the MIB Walk test case. The procedures using the "Get" approach shall be referred to as the "MIB Check" test case. Since the test procedures are equivalent, either one may be used to verify object instantiation.

1.8.3 User Guide

- 1.8.4 Operators Guide
- 1.8.5 Installation Guide

1.9 INPUT SPECIFICATIONS

The requirements as stated in NTCIP 1202 imply that the data elements within appropriate groups shall be instantiated. However, some of the groups and individual data elements are not mandatory.

In NTCIP 1202 v02.18 - Annex A, the table in Clause A.2 indicates whether a conformance group is mandatory or optional. The tables in Clauses A.3 through A.35 enumerate the objects in each conformance group and indicate which specific objects are mandatory or optional.

Note that the list does not add the ".0" in the case of leaf objects or the ".1 ... X" in the case of columnar objects. Annex B of this document contains a list of objects that contain max values. These objects serve to define the number of rows in various tables. The value of X in the ".1 ... X" extensions shall match the appropriate max value.

In the case of the MIB Walk test case, there are no input specifications per se. Get Next operations are performed until a noSuchName error is returned.

In the case of the MIB check test case, the input specification is a MIB compiler "tree output" containing all the potential objects that may be supported by the DUT.

1.10 OUTPUT SPECIFICATIONS

The output specification of object instantiation is a "GetResponse" with an Error-Status of noError when a GetRequest of OID is sent to the device under test. A GetResponse with an Error-Status of noSuchName indicates that the object is not supported and should be duly noted.

1.11 ENVIRONMENTAL NEEDS

1.11.1 Hardware

The following is the hardware needed to conduct this test case:

- Actuated Traffic Signal Controller that supports NTCIP 1202 Object Definitions,
- Test software that is capable of generating SNMP GetRequests and SetRequests with the OIDs and values as delineated above, and .
- Test software that is capable of parsing and displaying the values contained in an SNMP GetResponse.

Note - The term "Test software," as used above, is generic. The NTCIP Exerciser, SimpleSoft's SimpleTester™, or another software package that has the ability to generate the interpret SNMP Message may be used.

1.11.2 Software

The communications software in the DUT must be compliant with the communications used in the Test Software.

1.11.3 Other

The following is the configuration of the hardware components:



The use of a protocol analyzer is optional.

1.12 SPECIAL PROCEDURAL REQUIREMENTS

There are no special procedural requirements related to this test case.

1.13 INTERCASE DEPENDENCIES

There are no intercase dependencies related to this test case. The only requirement for reading data is use of the appropriate Community Name. To ensure access to all objects, the community name as defined in communityNameAdmin should be used.

Supported Values Test Case

2.0 TEST ITEMS

This test case applies to devices that implement the data elements defined within NTCIP 1202-ASC. While NTCIP 1202-ASC is, technically, an information profile that specifies all of the data elements to be implemented in an ASC device, this test case specification only deals with the objects that are defined in the MIB. All MIB defined objects are checked for instantiation and the range of values supported.

2.0.1 Requirement Specifications

The requirements specifications are defined in NTCIP 1202 and are summarized in NTCIP 1202 v02.19-Annex A. The following appears at the beginning of Annex A:

Conformance Groups are defined as either mandatory or optional. If a Conformance Group is mandatory, all of the objects with STATUS "mandatory" that are part of the Conformance Group shall be present for a device to claim conformance to the Conformance Group. If a Conformance Group is optional, all of the objects that are part of the Conformance Group with the STATUS "mandatory" shall be present if the device supports the Conformance Group. Objects with the STATUS "optional" may be supported.

When a table is included in a Conformance Group, all objects contained in the table are included by reference. This is because a table is defined as a SEQUENCE OF {SEQUENCE}. Thus, all objects listed in the sequence are defined as an integral part of the table. Tables are defined as either mandatory or optional. If a table is mandatory, all of the objects with STATUS "mandatory" shall be present. If a table is optional, all of the objects with the STATUS "mandatory" shall be present if the device supports the table. Objects in the table with the STATUS "optional" may be supported.

The following statements appear in Clause A.2:

Additional objects or groups may be supported without being non-compliant with ASC objects or NTCIP. Minimum and maximum ranges of objects that differ from the values of the object's SYNTAX field may be enforced by an application running on a device.

A device which enforces range limits within the bounds specified by the values of the object's SYNTAX field shall not be categorized as being non-compliant with ASC objects or NTCIP.

A device which supports a subset of objects with enumerated values shall not be categorized as being non-compliant with NTCIP 1202-ASC objects or NTCIP.

The table indicates that the individual data elements in the following conformance groups defined within NTCIP 1202 shall be supported:

- Phase Conformance Group
- Detector Conformance Group

The table also indicates that the individual data elements in the following conformance groups defined within NTCIP 1202 may be optional supported:

- Volume Occupancy Report Conformance Group
- Unit Conformance Group
- Special Function Conformance Group
- Coordination Conformance Group
- Time Base Conformance Group
- Preempt Conformance Group
- Ring Conformance Group

- Channel Conformance Group
- Overlap Conformance Group
- TS 2 Port 1 Conformance Group
- Block Object Conformance Group

It is assumed that test cases and procedures for the other Conformance Groups (and their data elements) are covered in other documents.

2.0.2 Design Specifications

- 2.0.3 User Guide
- 2.0.4 Operators Guide
- 2.0.5 Installation Guide

2.1 INPUT SPECIFICATIONS

The requirements as stated in NTCIP 1202 imply that the data elements within the appropriate group shall be instantiated and that the range or a subrange shall be supported.

A manufacturer's completed PICS should be included in the Test Item Transmittal Report. It should list the upper and lower bounds of an object's supported value range.

Annex B of this document contains some clarification on how to interpret what may be listed in a manufacturer's PICS.

2.2 OUTPUT SPECIFICATIONS

The output specification of the checks for supported values is a "GetResponse" with an Error-Status of noError when a "SetRequest" of the OID is sent to the device under test. Any other Error-Status indicates the value.

2.3 ENVIRONMENTAL NEEDS

2.3.1 Hardware

The following is the hardware needed to conduct this test case:

- Actuated Traffic Signal Controller that supports NTCIP 1202 Object Definitions.
- Test software that is capable of generating SNMP GetRequests and SetRequests with the OIDs and values as delineated above.
- Test software that is capable of parsing and displaying the values contained in an SNMP GetResponse.

2.3.2 Software

The communications software in the DUT must be compliant with that used in the Exerciser.

2.3.3 Other

The following is the configuration of the hardware components:



The use of a protocol analyzer is optional.

2.4 SPECIAL PROCEDURAL REQUIREMENTS

There are no special procedural requirements related to this test case.

2.5 INTERCASE DEPENDENCIES

The intercase dependency related to this test case specification is support of the Database Management Conformance Group whose data elements are defined in NTCIP 1201. Testing of "P2" Object Types requires support of dbCreateTransaction.

SNMP Protocol Test Case

3.0 TEST ITEMS

The overall purpose of this test case is to verify conformance to the SNMP Protocol. The SNMP items and features to be exercised by this test case include:

- getNextRequest
- getRequest
- setRequest
- getResponse
- Multiple Variable Binding in a setRequest
- Multiple Variable Binding in a getRequest
- Errors
 - o badValue
 - o **readOnly**
 - o noSuchName
 - o badValue in multiple variable binding
- Invalid communityName
- Data Elements in SNMP Conformance Group

3.1 INPUT SPECIFICATIONS

The inputs specifications for this test case are defined in the General, Get and Set Commands, SNMP Errors, Encoding Rules, and SNMP Configuration Group Sessions of the VDOT- SNMP Test Procedures Report - Rev 1.doc.

3.2 OUTPUT SPECIFICATIONS

The output specifications for this test case are defined in the General, Get and Set Commands, SNMP Errors, Encoding Rules, and SNMP Configuration Group Sessions of the VDOT- SNMP Test Procedures Report - Rev 1.doc.

3.3 ENVIRONMENTAL NEEDS

3.3.1 Hardware

The following is the hardware needed to conduct this test case:

- Actuated Traffic Signal Controller that supports the SNMP portions of NTCIP 2301 Simple Transportation Management Framework Application Profile.
- Test software that is capable of generating SNMP GetRequests and SetRequests with the OIDs and values as delineated above.
- Test software that is capable of parsing and displaying the values contained in an SNMP GetResponse.

3.3.2 Software

The communications software in the DUT must be compliant with that used in the Exerciser.

3.3.3 Other

The following is the configuration of the hardware components:



The use of a protocol analyzer is optional.

3.4 SPECIAL PROCEDURAL REQUIREMENTS

There are no specific special procedural requirements associated with this test case.

3.5 INTERCASE DEPENDENCIES

There are no specific intercase dependencies associated with this test case.

Null Protocol Test Case

4.0 TEST ITEMS

The items and features to be exercised by this test case include:

• Handling of Initial Protocol Identifier

4.1 INPUT SPECIFICATIONS

The inputs specifications for this test case are defined in HDLC Errors Session of the VDOT- SNMP Test Procedures Report - Rev 1.doc.

4.2 OUTPUT SPECIFICATIONS

The output specifications for this test case are defined in HDLC Errors Session of the VDOT- SNMP Test Procedures Report - Rev 1.doc.

4.3 ENVIRONMENTAL NEEDS

4.3.1 Hardware

The following is the hardware needed to conduct this test case:

- Actuated Traffic Signal Controller that supports NTCIP 1202 Object Definitions.
- Test software that is capable of generating SNMP GetRequests and SetRequests with the OIDs and values as delineated above.
- Test software that is capable of parsing and displaying the values contained in an SNMP GetResponse.

4.3.2 Software

The communications software in the DUT must be compliant with that used in the Exerciser.

4.3.3 Other

The following is the configuration of the hardware components:



The use of a protocol analyzer is optional.

4.4 SPECIAL PROCEDURAL REQUIREMENTS

There are no specific special procedural requirements associated with this test case.

4.5 INTERCASE DEPENDENCIES

There are no specific intercase dependencies associated with this test case.

PMPP Using RS-232 Test Case

5.0 TEST ITEMS

The items and features to be exercised by this test case include:

- address field
 - o one byte form
 - two byte form
- broadcast
- group address
 - o one byte form
 - o two byte form
- control field
- HDLC errors
 - o unknown IPI
 - o Invalid CRC value
 - o Invalid data stream
- data elements in RS-232 Conformance Group
- data elements in LapB Conformance Group

5.1 INPUT SPECIFICATIONS

The input specifications for this test case are defined in VDOT- Class B Test Procedures - Rev 1.doc.

5.2 OUTPUT SPECIFICATIONS

The output specifications for this test case are defined in VDOT- Class B Test Procedures - Rev 1.doc.

5.3 ENVIRONMENTAL NEEDS

5.3.1 Hardware

The following is the hardware needed to conduct this test case:

- Actuated Traffic Signal Controller that supports NTCIP 1202 Object Definitions.
- Test software that is capable of generating SNMP GetRequests and SetRequests with the OIDs and values as delineated above.
- Test software that is capable of parsing and displaying the values contained in an SNMP GetResponse.

5.3.2 Software

The communications software in the DUT must be compliant with that used in the Exerciser.

5.3.3 Other

The following is the configuration of the hardware components:



The use of a protocol analyzer is optional.

5.4 SPECIAL PROCEDURAL REQUIREMENTS

There are no specific special procedural requirements associated with this test case.

5.5 INTERCASE DEPENDENCIES

There are no specific intercase dependencies associated with this test case.

PMPP Using FSK Modem Test Case

The tests conducted with respect to the Point-to-Multipoint Protocol using an FSK Modem Interface shall consist of conducting the System Operational Scenarios with a 1200 Bps FSK Modem (see Subclause 1.4.7). This test is contingent upon DUT support of the NTCIP 2102 (PMPP using FSK Modem).

6.0 TEST ITEMS

The items and features to be exercised by this test case include:

• FSK Modem

6.0.1 Requirements specifications

Execution of this test case requires the DUT to support an FSK Modem interface.

6.1 INPUT SPECIFICATIONS

6.2 OUTPUT SPECIFICATIONS

6.3 ENVIRONMENTAL NEEDS

6.3.1 Hardware

The following is the hardware needed to conduct this test case:

- Actuated Traffic Signal Controller that supports NTCIP 1202 Object Definitions and an FSK Modem Interface.
- An external FSK Modem to be connected to the test software computer.
- Test software that is capable of generating SNMP GetRequests and SetRequests with the OIDs and values as delineated above.
- Test software that is capable of parsing and displaying the values contained in an SNMP GetResponse.

6.3.2 Software

The communications software in the DUT must be compliant with that used in the Exerciser.

6.3.3 Other

The following is the configuration of the hardware components:



The use of a protocol analyzer is optional.

6.4 SPECIAL PROCEDURAL REQUIREMENTS

There are no specific special procedural requirements associated with this test case.

6.5 INTERCASE DEPENDENCIES

There are no specific intercase dependencies associated with this test case.

System Operational Scenarios Test Cases

7.0 TEST ITEMS

The items and features to be exercised by this test case include:

- Setting and verifying the time and date
- Setting up several Dynamic Objects
- Retrieving data typically used to display an intersection map
- Retrieving the status of an eight intersection system
- Two channel operation

7.0.1 Requirements specifications

- A. For setting and verifying the time and date, the following NTCIP 1202 objects will be checked for proper operation:
 - globalTime
 - controller-localTime
 - globalDaylightSavings
 - controller-standardTimeZone
- B. For setting up several Dynamic messages, two dynamic objects will be defined that correspond to the objects used in the following:
 - intersection map
 - eight intersection system
- C. To simulate an eight-phase intersection map, the following NTCIP 1202 objects will be retrieved:
 - channelStatusGroupGreens (Group 1)
 - channelStatusGroupYellows (Group 1)
 - vehicleDetectorStatusGroupActive (Group 1)
 - phaseStatusGroupPedCall (Group 1)
 - cordPatternStatus
 - unitControlStatus
 - shortAlarmStatus
 - ringStatus (sequenceRingNumber 1&2)

If the DUT supports more than 8 phases, the following objects will be retrieved:

- channelStatusGroupGreens (Groups 1&2)
- channelStatusGroupYellows (Groups 1&2)
- vehicleDetectorStatusGroupActive (Groups 1&2)
- phaseStatusGroupPedCalls (Group 1)
- cordPatternStatus
- unitControlStatus
- shortAlarmStatus
- ringStatus (sequenceRingNumber 1&2)
- D. For the status of an eight-intersection system, the following NTCIP 1202 objects will be retrieved from each intersection:
 - channelStatusGroupGreens (Group 1)
 - vehicleDetectorStatusGroupActive (Group 1)
 - shortAlarmStatus

- E. Two-channel operation is a test of a management application and therefore will not be included in this test case.
- 7.0.2 Design Specifications
- 7.0.3 User Guide
- 7.0.4 Operators Guide
- 7.0.5 Installation Guide
- 7.1 INPUT SPECIFICATIONS

7.2 OUTPUT SPECIFICATIONS

7.3 ENVIRONMENTAL NEEDS

7.3.1 Hardware

The following is the hardware needed to conduct this test case:

- Actuated Traffic Signal Controller that supports NTCIP 1202 Object Definitions.
 - Management software that is capable of generating SNMP GetRequests and SetRequests with the OIDs and values as delineated above.
- Management software that is capable of parsing and displaying the values contained in an SNMP GetResponse.

Note - The term "Management software," as used above, is generic. It refers to any PC Software package that has the ability to generate and interpret SNMP messages in the manner described above.

7.3.2 Software

The communications software in the DUT must be compliant with that used in the Management Software.

7.3.3 Other

The following is the configuration of the hardware components:



The use of a protocol analyzer is optional.

7.4 SPECIAL PROCEDURAL REQUIREMENTS

There are no specific special procedural requirements associated with this test case.

7.5 INTERCASE DEPENDENCIES

There are no specific intercase dependencies associated with this test case.

Optional System Operational Scenarios Test Cases - Retrieving Log Data

8.0 TEST CASE SPECIFICATION IDENTIFICATION

8.1 TEST ITEMS

8.1.1 Requirements Specifications

A. For setting and verifying the time and date, the following NTCIP 1202 objects will be checked for proper operation:

maxEventClasses eventClassTable eventClassEntry eventClassNumber eventClassLimit eventClassClearTime eventClassDescription eventClassNumRowsInLog eventClassNumEvents maxEventLogConfigs eventLogConfigTable eventLogConfigEntry eventConfigID eventConfigClass eventConfigMode eventConfigCompareValue eventConfigCompareValue2 eventConfigCompareOID eventConfigLogOID eventConfigAction eventConfigStatus maxEventLogSize eventLogTable eventLogEntry eventLogClass eventLogNumber eventLogID eventLogTime eventLogValue numEvents

8.1.2 Design Specifications

A. The following are the pre-conditions:

- maxEventClasses shall support a minimum value of 6.
- The controller is configured a standard 8-phase quad.
- Controller is cycling through all 8 phases and cycle time should be 100 seconds.
- All eventConfigAction.X are set = disabled(2).
- 5 < phaseMinimumGreen.1 > 15.
- B. Configure the time management entries as follows:

globalTime. 0	globalDaylightSaving.0	controller-standardTimeZone.0
local time + 21600	enableUSDST	- 21600 (CST = 6 hours)

Note: This sets controller-localTime to current TOD.

eventClassNumbe	eventClassLimi	eventClassClearTime	eventClassDescription
r	t		
1	2	controller-localTime	"Class 1, Limit 2, onChange"
2	3	controller-localTime	"Class 2, Limit 3, greaterThan"
3	4	controller-localTime	"Class 3, Limit 4, smallerThan"
4	5	controller-localTime	"Class 4, Limit 5, hystereis"
5	6	controller-localTime	"Class 5, Limit 6, periodic"
6	7	controller-localTime	"Class 6, Limit 7, andWith"

C. Configure the entries in the eventClassTable as follows:

Note: Setting eventClassClearTime = controller-localTime clears any previous events

D. Configure the entries in eventLogConfigTable as follows:

			1	1			
ID	Class	Mode	Compare Value	Compare Value2	CompareOID	Log OID	Action
1	1	onChange(2)	n/a	n/a	phaseMinimumGreen.1	phaseMinimum Green.1	disabled (2)
2	2	greaterThan(3)	0x44	10	phaseStatusGroupGreens.1	phaseStatus GroupGreens.1	disabled (2)
3	3	smallerThan (4)	0x88	10	phaseControlGroupHold.1	phaseControl GroupHold.1	disabled (2)
4	4	hysteresis (5)	6	7	phaseMinimumGreen.1	phaseMinimum Green.1	disabled (2)
5	5	Periodic (6)	10	n/a	globalTime.0	coordCycle Status.0	disabled (2)
6	6	andedWith (7)	0x11	n/a	phaseStatusGroupPhase Ons.1	phaseStatus GroupPhase Ons.1	disabled (2)

Note: After configuration and having the controller complete 2 complete cycles, is a check of the logs should indicate that they are cleared.

E. For Class 1 Events, configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare Value	Compare Value2	CompareOID	Log OID	Action
1	1	onChange(2)	n/a	n/a	phaseMinimumGreen.1	phaseMinimum Green.1	log (2)

Then perform the following:

Set phaseMinimumGreen.1= phaseMinimumGreen.1 + 1 Wait 15 seconds Set phaseMinimumGreen.1= phaseMinimumGreen.1 Verify eventClassNumEvents.ClassNumber = 2 Verify numEvents.0 = 2 Verify that:

eventLogClas	eventLogNumbe	eventLogID	eventLogTime	eventLogValue
S	r			
1	1	1	~current local Time - 15	phaseMinimumGreen. 1 + 1
1	2	1	~current local Time	phaseMinimumGreen. 1

Configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare Value	Compare Value2	CompareOID	Log OID	Action
1	1	onChange(2)	n/a	n/a	phaseMinimum Green.1	phaseMinimum Green.1	disabled(2)

Configure the entries in the eventClassTable as follows:

eventClassNumber	eventClassLimit	eventClassClearTime	eventClassDescription
1	2	current-localTime	"Class 1, Limit 2, onChange"

Verify eventClassNumEvents.1 = 0

F. For Class 2 Events, configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare Value	Compare Value2	CompareOID	Log OID	Action
2	2	greaterThan(3)	0x44	10	phaseStatusGroup Greens.1	phaseStatus GroupGreens.1	log (3)

Then perform the following: Wait 330 Seconds Verify eventClassNumEvents.2 = 3 Verify that:

eventLogClas	eventLogNumbe	eventLogID	eventLogTime	eventLogValue
S	r			
2	1	2		0x88
2	2	2	> eventLogTime.1	0x88
2	3	2	> eventLogTime.2	0x88

Configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare	Compare	CompareOID	Log	Action
			Value	Value2		OID	
2	2	greaterThan(3)	0x44	2	phaseStatusGroup Greens.1	phaseStatusGroup Greens.1	Disabled (2)

Configure the entries in the eventClassTable as follows:

eventClassNumber	eventClassLimit	eventClassClearTime	eventClassDescription
2	3	current-localTime	"Class 2, Limit 3,
			greaterThan"

Verify eventClassNumEvents.2 = 0

G. For Class 3 Events, configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare Value	Compare Value2	CompareOID	Log OID	Action
3	3	smallerThan (4)	0x01	10	phaseControlGroup Hold.1	phaseControlGroup Hold.1	log (3)

Then perform the following:

Wait 15 seconds
Set phaseControlGroupHold.1 = 0x01 Wait 15 seconds Set phaseControlGroupHold.1 = 0x00 Wait 15 seconds Verify eventClassNumEvents.3 = 3 Verify that:

eventLogClas	eventLogNumbe	eventLogID	eventLogTime	eventLogValue
S	r			
3	1	3	~current local Time -35	0x00
3	2	3	~current local Time -20	0x01
3	3	3	~current local Time – 5	0x00

Configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare Value	Compare Value2	CompareOID	Log OID	Action
3	3	smallerThan (4)	0x88	10	phaseControlGroup Hold.1	phaseControlGroup Hold.1	disabled(2)

Configure the entries in the eventClassTable as follows:

eventClassNumber	eventClassLimit	eventClassClearTime	eventClassDescription
3	4	current localTime	"Class with limit of 4."

Verify eventClassNumEvents.3 = 0

H. For Class 4 Events, configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare Value	Compare Value2	CompareOID	Log OID	Action
4	4	hysteresis (5)	5	15	phaseMinimum Green.1	phaseMinimum Green.1	log (3)

Then perform the following:

Set phaseMinimumGreen.1= 0x04 Wait 15 Seconds

Set phaseMinimumGreen.1= 0x10 Wait 15 Seconds

Set phaseMinimumGreen.1= original value

Wait 15 seconds

Verify eventClassNumEvents.3 = 2

Verify that:

eventLogClas	eventLogNumbe	eventLogID	eventLogTime	eventLogValue
S	r			
4	1	4	~current local Time -	0x04
			20	
4	2	4	~current local Time - 5	0x10

Configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare Value	Compare Value2	CompareOID	Log OID	Action
4	4	hysteresis (5)	6	7	phaseMinimum Green.1	phaseMinimum Green.1	disabled(2)

Configure the entries in the eventClassTable as follows:

eventClassNumber	eventClassLimit	eventClassClearTime	eventClassDescription
4	4	current localTime	"Class with limit of 5."

Verify eventClassNumEvents.4 = 0

I. For Class 5 Events, configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare Value	Compare Value2	CompareOID	Log OID	Action
5	5	periodic (6)	10	n/a	globaltime.0	coordCycle Status.0	log (3)

Then perform the following: Wait 125 seconds

Verify eventClassNumEvents.3 = 6 Verify that:

eventLogClas	eventLogNumbe	eventLogI	eventLogTime	eventLogValue
S	r	D	-	_
5	1	5	~current local Time - 55	~current local Time - 55
5	2	5	~current local Time - 45	~current local Time - 45
5	3	5	~current local Time - 35	~current local Time - 35
5	4	5	~current local Time - 25	~current local Time - 25
5	5	5	~current local Time - 15	~current local Time - 15
5	6	5	~current local Time - 5	~current local Time - 5

Configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare Value	Compare Value2	CompareOID	Log OID	Action
5	5	periodic (6)	10	n/a	globalTime.0	coordCycle Status.0	disabled(2)

Configure the entries in the eventClassTable as follows:

eventClassNumber	eventClassLimit	eventClassClearTime	eventClassDescription
5	4	current localTime	"Class with limit of 5."

Verify eventClassNumEvents.4 = 0

J. For Class 6 Events, configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare Value	Compare	CompareOID		Action
6	6	andedWith (7)	0x11	n/a	phaseStatusGroup PhaseOns.1	phaseStatusGroup PhaseOns.1	log (3)

Then perform the following: Wait 5 minutes Verify eventClassNumEvents.6 = 6 Verify that:

eventLogClas	eventLogNumbe	eventLogID	eventLogTime	eventLogValue
S	r			
6	1	6	n/a	0x11
6	2	6	n/a	0x11
6	3	6	n/a	0x11
6	4	6	n/a	0x11
6	5	6	n/a	0x11
6	6	6	n/a	0x11

Configure the entries in eventLogConfigTable as follows:

ID	Class	Mode	Compare	Compare	CompareOID	Log	Action
			Value	Value2		OID	
6	6	andedWith (7)	0x11	n/a	phaseStatusGroup PhaseOns.1	phaseStatusGroup PhaseOns.1	disabled (3)

Configure the entries in the eventClassTable as follows:

eventClassNumber	eventClassLimit	eventClassClearTime	eventClassDescription
6	6	current localTime	"Class with limit of 6."

Verify eventClassNumEvents.4 = 0

Note that the test case of retrieving log data requires support of the NTCIP 1202 v2 Block Object Conformance Groups and the Block Data Type and ID requirements.

Optional System Operational Scenarios Test Cases - Retrieving Log Data

9.0 TEST CASE SPECIFICATION IDENTIFICATION

The test case of setting up a timebase schedule of events requires support of the NTCIP 1202 v2 TimeBlock Object Conformance Groups and the Block Data Type and ID requirements.

The test case of performing the database upload and download requires support of the NTCIP 1202 v2 Block Object Conformance Groups and the Block Data Type and ID requirements.

- 9.0.1 Design Specifications
- 9.0.2 User Guide
- 9.0.3 Operators Guide
- 9.0.4 Installation Guide
- 9.1 INPUT SPECIFICATIONS
- 9.2 OUTPUT SPECIFICATIONS

9.3 ENVIRONMENTAL NEEDS

9.3.1 Hardware

The following is the hardware needed to conduct this test case:

- Actuated Traffic Signal Controller that supports NTCIP 1202 Object Definitions.
- Management Software that is capable of generating SNMP GetRequests and SetRequests with the OIDs and values as delineated above.
- Management Software that is capable of parsing and displaying the values contained in an SNMP GetResponse.

9.3.2 Software

The communications software in the DUT must be compliant with that used in the Exerciser.

9.3.3 Other

The following is the configuration of the hardware components:



The use of a protocol analyzer is optional.

9.4 SPECIAL PROCEDURAL REQUIREMENTS

There are no specific special procedural requirements associated with this test case.

9.5 INTERCASE DEPENDENCIES

There are no specific intercase dependencies associated with this test case.

System Performance Testing Test Case

10.0 TEST CASE SPECIFICATION IDENTIFICATION

10.1 TEST ITEMS

The items and features to be exercised by this test case include:

- Operation of Six-Intersection System using 1200 BPS
- Operation of Six-Intersection System using 9600 BSP

One or more test cases will be defined to maximize the number of messages that can be exchanged when the Subnetwork operates at 1200 BPS. The mix of messages should include single addressed gets and sets and one or more group addressed sets. Given the throughput limitations of 1200 BPS, the STMP Protocol will be used.

One or more test cases will be defined to maximize the number of messages that can be exchanged when the Subnetwork operates at 9600 BPS. The mix of messages should include single addressed gets and sets and one or more group addressed sets. Both the SNMP and STMP Protocols will be used to conduct these tests.

10.1.1 Requirements specifications

One or more test cases will be defined to maximize the number of messages that can be exchanged when the Subnetwork operates at 1200 BPS. The mix of messages should include single addressed gets and sets and one or more group addressed sets. Given the throughput limitations of 1200 BPS, the STMP Protocol will be used.

One or more test cases will be defined to maximize the number of messages that can be exchanged when the Subnetwork operates at 9600 BPS. The mix of messages should include single addressed gets and sets and one or more group addressed sets. Both the SNMP and STMP Protocols will be used to conduct these tests.

10.1.2 Design Specifications

The message used in the Phoenix and Lakewood system will be used:

- Once per second status poll (long and short)
- Command timing plan and special functions
- Upload/download system time and DST flag
- Polling for Detector volume and Occupancy

10.1.3 User Guide

- 10.1.4 Operators Guide
- 10.1.5 Installation Guide

10.2 INPUT SPECIFICATIONS

NTCIP 1202 does not define specific requirements in terms of throughput and response time. However, the initial installation of the NTCIP Conformant Traffic Signal Controllers provides some guidance on a systems integrator's expectation. Transcore's "NTCIP Lessons or Making NTCIP Work" (see Lessons Learned Transcore v1.doc) provides specifics about typical messages and timing.

10.3 OUTPUT SPECIFICATIONS

10.4 ENVIRONMENTAL NEEDS

10.4.1 Hardware

The following is the hardware needed to conduct this test case:

- Actuated Traffic Signal Controller that supports NTCIP 1202 Object Definitions.
- Management software that is capable of generating SNMP GetRequests and SetRequests with the OIDs and values as delineated above.
- Management software that is capable of parsing and displaying the values contained in an SNMP GetResponse.
- A suitable modem or RS-232 converters capable of simulating multi-drop.

Depending on the configuration of the DUT, an external modem may be required.

10.4.2 Software

The communications software in the DUT must be compliant with that used in the Exerciser.

10.4.3 Other

The following are two possible configurations for the hardware components of this test case. Figure 1 represents the case where the Traffic Signal Controllers all share a common multi-drop interface.





Figure 2 represents an alternate were the RS-232 Interface on the Traffic Signal Controllers is used but converted to multi-drop by some means such as RS-232 to RS-485 Converters.



Figure 2

The use of a protocol analyzer is optional.

10.5 SPECIAL PROCEDURAL REQUIREMENTS

There are no specific special procedural requirements associated with this test case.

10.6 INTERCASE DEPENDENCIES

There are no specific intercase dependencies associated with this test case.

ASC Object Identifiers and Names

The following is a list of the Object Identifiers and Object Names that are defined or referenced in NTCIP 1202v02.19.

<Include the latest mib.out file from the SMICng Compiler>

ASC Max Value Object Identifiers

The following is a list of the Object Names defined or referenced in NTCIP 1202 that specify the upper bound or max values for various table indexes. These objects serve to define the number of rows in various tables. For columnar objects, the maximum value of the instance identifiers added to the Object Identifiers for columnar objects shall not exceed the max value.

The following max values are defined in NTCIP 1202:

- maxPhases
- maxPhaseGroups
- maxVehicleDetectors
- maxVehicleDetectorStatusGroups
- maxPedestrianDetectors
- maxAlarmGroups
- maxSpecialFunctionOutputs
- maxPatterns
- maxSplits
- maxTimeBaseScheduleEntries
- maxDayPlans
- maxDayPlanEvents
- maxTimebaseAscActions
- maxPreempts
- maxRings
- maxSequences
- maxRingControlGroups
- maxChannels
- maxChannelStatusGroups
- maxOverlaps
- maxOverlapStatusGroups
- maxPort1Addresses

The following max values are defined in NTCIP 1201 - Global Object Definitions:

- globalMaxModules
- maxTimeBaseScheduleEntries
- maxDayPlans
- maxDayPlanEvents
- maxEventLogConfigs
- maxEventLogSize
- maxEventClasses
- maxGroupAddress

The following max values are defined in NTCIP 1103 - Transportation Management Protocol:

- logicalNameTranslationTable-maxEntries
- communityNamesMax

The following max values are defined in NTCIP 2103 - PPP Subnetwork Profile:

chapMaxSecrets

The following max values are defined in NTCIP 2202 (RFC 1213) - Internet Transport Profile:

- tcpMaxConn
- ifNumber

The following max values are defined in NTCIP 2101 and 2102 (RFC 1317) - PMPP with RS-232 and FSK Subnetwork Profiles:

• rs232Number

Note: A number of SNMPv1 Interface MIBs do not follow the MIB-II (RFC 1213) convention for identifying table indexes that contain parameters associated with interfaces. Basically, MIB-II assumes that the maximum value of ifIndex is the total number of physical interfaces supported. The value of ifIndex is used to point to one and only one physical interface.

For example, a hypothetical traffic signal controller could have three interfaces that support NTCIP protocols. It might have a RS-232 Interface as its system communications port, an RS-485 Interface as its intra-cabinet communications port, and an Ethernet Interface as its console or laptop communications port. The functionally of the RS-232 and RS-485 interfaces is expressed in RFC 1317 - Definitions of Managed Objects for RS-232-like Hardware Devices. The functionality of the Ethernet Interface is expressed in RFC 1643 - Definitions of Managed Objects for Ethernet-like Interface Types.

In the Ethernet MIB, ifIndex is used as the index in various tables that point to parameters associated with that type of interface. There is no object definition that defines the total number of Ethernet interfaces. In the RS-232 MIB, however, rs232Number defines the total number of ports covered by the MIB and is used as the index into various tables. When RS-232 MIB was re-written to support SNMPv2, however an object still defines the number of ports covered by the MIB but the index into the various tables was changed to ifIndex.

What this means is that even though there could be two RS-232-like interfaces in the hypothetical traffic signal controller, the ifIndex 1 could point to the RS-485 Interface, ifIndex 2 could point to the Ethernet Interface, and ifIndex 3 could point to the RS-232 Interface. There is no NTCIP object definition that maps the ifIndex to the rs232Number. Unless the definition of rs232PortIndex is revised, the value used could be confused with the value of ifIndex.

REFERENCES FOR APPENDIX F

- *1.* IEEE Std 829-1998 IEEE Standard for Software Test Documentation, Institute of Electrical and Electronics Engineers, New York, New York, 1998.
- DMS-11170, Fully Actuated, Solid-State Traffic Signal Controller Assembly, Departmental Material Specification, Texas Department of Transportation, August 2006, <u>http://manuals.dot.state.tx.us/dynaweb/colmates/dms/@ebt-link;?target=idmatch(s070019)</u>. Accessed July 29, 2006.
- 3. NTCIP Laboratory Testing for Actuated Signal Controllers, Summary Report for ASSHTO Project 475070. Published by Texas Transportation Institute. <u>http://tti.tamu.edu/documents/TTI-2006-1.pdf</u>. Accessed June 7, 2006.
- 4. ISO 9000 and ISO 14000 in brief, ISO 9000, <u>http://www.iso.org/iso/en/iso9000-14000/understand/inbrief.htm</u>. Accessed August 16, 2006.

APPENDIX G: TRAFFIC SIGNAL CONTROLLER TEST PROCEDURES

INTRODUCTION

This appendix contains a limited set of test procedures for use with NTCIP conformant traffic signal controller field devices. One intent of this research project is to develop a limited set of test procedures for ITS field devices using NTCIP standards to provide cost and effort estimates for creating them for devices in the remaining standardized application areas. Due to the scope and complexity of procedures for traffic signal controllers, the researcher's development focused on procedures that directly relate to TxDOT requirements that would not be part of any outside development. Per agreement with the project monitoring committee, the focus of the research was on testing procedures for the four-phase diamond sequencing and detector operations. Only the details of the prequalification, the four-phase diamond sequencing, and the four-phase detector operations appear after the test case summary. Table G-1 summarizes the test procedure that are a part this research effort.

Table G-2 summarizes some additional test procedures that may be applicable to traffic signal controller testing. However, their development was part of an AASHTO research project (*1*). The Test Design Specification and Test Case Specification in Appendix F includes general background and planning information that relates to development of the these test procedures.

Detector Operations

It is worthy to comment on two issues with respect to the detector operations test procedures. The first issue is that NTCIP does not provide a means to activate detector inputs prior to the operations in a controller that may operate on them. The NTCIP 1202-ASC object phaseControlGroupVehCall places a call for service on a particular phase, but there is no object that acts as a control to activate a detector input (2). Detector inputs may undergo memory, switching, delay, and extending operations prior to their routing to a phase. Since the detector operations tests involve phase switching, automating the testing process would be impossible without a means of activating detector inputs through software. A test script interface to the Hardware-in-the-Loop (HITL) software resolves the issue (*3*).

A "Set HITL Detector Input X = On/Off" instruction in the test procedures, sends an operating system inter-processor communications command to the HITL software to activate or

deactivate a detector input. The HITL software, in turn, sends the command to the software that controls a TS2 Test Box. The TS2 Test Box uses the Port 1 communication interface of a controller to activate the detector input of the traffic signal controller. There are delays and timing issues to consider when using a HITL interface, but it does provide a solution. The potential for using HITL in testing to provide independent verification of controller operation is worthy of further research. One does not have to rely on visual inspection or retrieval of NTCIP status information to confirm that some action takes place.

The second issue with the detector operations test procedures relates to risk management. A requirement for most of the four-phase diamond detectors is that they operate as both calling and extending detectors. In this operating mode, a detector extends, for example phase 1, when phase 1 is green. The detector also places a call on phase 1 when it is not in phase 1 green. Not in phase 1 means that it could be in phase 2, 3, 4, 5, 6, 7, or 8, for example. Rather than developing a test procedure that samples only some of the other possible states, the test procedures in this appendix look at all of the other intervals. In the case of the four-phase diamond operation, there are 12 distinct states. To ensure that 10 calling detectors operate correctly in the 11 other possible states requires 110 tests. If one considers that a phase can be in one of three possible substates (green, yellow, and red), there could be 330 possible states.

For the purpose of this research project, the researcher chose the more extensive 110-test approach rather that a sampling approach. One test procedure addresses the substates but the rest look at operation during the green only. The number of resulting test steps illustrates the impact of full coverage testing. Of the 183 pages needed to document the 110-test approach, a test procedure that chooses just one of the possible 11 phase states randomly would have reduced the documentation and resulting effort to implement the tests by at least 75%.

A sampling approach runs the risk of not detecting a potential problem in some situations. The probability is that in the long term, it would show up after some number of testing sessions. As it turns out, the extensive testing approach did uncover a problem immediately. Two detectors fail to call the designated phase when in the green of one particular other phase. Both detectors operate per the requirements. The physical geometry of an intersection or signage may preclude the need to call the designated phase in this particular situation. In the testing, however, it does not appear to be correct. A recommendation of the research is to investigate this further.

248

TEST CASE SUMMARY

Table G-1 provides a summary of the traffic signal controller test procedures and test cases organized by feature or functional area derived from NTCIP 1202-ASC and DMS 11170-TSC (2,4). Two procedures, Global Configuration and Security, have a reference in NTCIP 1202-ASC, but the object definitions appear in NTCIP 1201-GLO (2,5). The test procedures do not address all the functional requirements in DMS 11170-TSC (4).

	Traffic Signal Controller Test Cases			
ID	Title	Description		
Prequa	lification (PRL)			
TC001	ASC PRL Information	This procedure retrieves minimum project requirements and maximum values, checks for whether the required objects are implemented, and performs a sampling of the supported values.		
Four-P	hase Diamond Sequencing (Seq)			
TC001	Sequencing	Tests sequencing of all transitions from one state to another using vehicle calls (not detector calls).		
E D				
Four-P	hase Diamond Detector Operation	ons (DetOps)		
1C001	Detector I Operations	Verifies the operation of Detector 1 to call Phase 6 under specific conditions and extend intervals 2516B, 2517B, 2518B, 4517B, 4518B, 1517B, and 3518B.		
TC002	Detector 2 Operations	Verifies the operation of Detector 2 to call and extend Phase 2. The test for the two-second delay is in TC021.		
TC003	Detector 3 Operations	Verifies the operation of Detector 3 to call and extend Phase 3 under specific conditions and to extend interval 3516B. The test for the two-second delay is in TC021.		
TC004	Detector 4 Operations	Verifies the operation of Detector 4 to call and extend Phase 4 under specific conditions and extend interval 4516B. The test for the two-second delay is in TC021.		
TC005	Detector 5 Operations	Verifies the operation of Detector 5 to call Phase 2 under specific conditions and extend intervals 1625B, 1635, 1645B, 1735B, 1745B, 1835B, and 1845B.		

Table G-1. Traffic Signal Controller Test Case Summary.

Traffic Signal Controller Test Cases			
ID	Title	Description	
TC006	Detector 6 Operations	Verifies the operation of Detector 6 to call and extend Phase 6. The test for the two-second delay is in TC021.	
TC007	Detector 7 Operations	Verifies the operation of Detector 7 to call and extend Phase 7 under specific conditions and extend interval 1725B. The test for the two-second delay is in TC021.	
TC008	Detector 8 Operations	Verifies the operation of Detector 8 to call and extend Phase 8 under specific conditions and extend interval 1825B. The test for the two-second delay is in TC021.	
TC009	Detector 9 Operations	Verifies the operation of Detector 9 to call Phase 6 under specific conditions, extend Phase 2 under specific conditions, and extend intervals 2516B, 2517B, 2518B, 3517B, 3518B, 4517B, and 4518B.	
TC010	Detector 10 Operations	Verifies the operation of Detector 10 to call Phase 6 under specific conditions, extend Phase 2 under specific conditions, and extend intervals 2516B, 2517B, 2518B, 3517B, 3518B, 4517B, and 4518B.	
TC011	Detector 11 Operations	Verifies the operation of Detector 11 to call and extend Phase 2 under specific conditions	
TC012	Detector 12 Operations	Verifies the operation of Detector 12 to call and extend Phase 4 under specific conditions and extend interval 4516B (6+12).	
TC013	Detector 13 Operations	Verifies the operation of Detector 13 to call Phase 2 under specific conditions, extend Phase 6 under specific conditions, and extend intervals 1625B, 1635B, 1645B, 1735B, 1745B, 1835B, and 1845B.	
TC014	Detector 14 Operations	Verifies the operation of Detector 14 to call Phase 2 under specific conditions, extend Phase 6 under specific conditions, and extend intervals 1625B, 1635B, 1645B, 1735B, 1745B, 1835B, and 1845B.	
TC015	Detector 15 Operations	Verifies the operation of Detector 15 to call and extend Phase 6 under specific conditions.	
TC016	Detector 16 Operations	Verifies the operation of Detector 16 to call and extend Phase 8 under specific conditions and extend interval 1825B.	
TC017	Detector 17 Operations	Verifies the operation of Detector 17 to call and extend Phase 3 under specific conditions and extend interval 3516B.	

Table G-1. Traffic Signal Controller Test Case Summary (continued).

	Traffic Signal Controller Test Cases			
ID	Title	Description		
TC018	Detector 18 Operations	Verifies the operation of Detector 18 to call and extend Phase 7 under specific conditions and extend interval 1725B.		
TC019	Detector Operations Setup	This procedure performs general setup of controller parameters to facilitate testing and provide consistent operation.		
TC020	Detector Operations Teardown	This procedure restores original controller parameters after executing Detector Operation Setup – TC001.		
TC021	Detector Delay	Verifies that, when programmed, Detectors 2, 3, 4, 6, 7, and 8 delay entering a call for the parent phase when the parent phase is red.		
Global	Configuration (GloCon)			
TC001	Retrieve Module Table	This procedure retrieves the module table, and allows the Tester to verify that the device under test (DUT) reports the proper type of device, manufacturer, model, and version.		
TC002	Global Set ID	This procedure ensures that a change to a static database object produces a change in globalSetIDParmeter.		
Security	y			
TC001	Change Administrator Community Name	Verifies that the administrator can change the administrator community name stored in the DUT and properly affects operations.		
TC002	Change User Community Name	Verifies that the administrator can change the user community names and their masks stored in the DUT and properly affects operations.		

Table G-1. Traffic Signal Controller Test Case Summary (continued).

The details of the Global Configuration and Security test cases can be found in Appendix C of this report.

Some additional test procedures that may be applicable to traffic signal controller testing are summarized in Table G-2. However, these test procedures were developed as part of an AASHTO research project (1). The details of each procedure and the associated test scripts are available at <u>www.itstestlab.org</u>. The Test Design Specification and Test Case Specification in Appendix F include discussion on the premise and background information on these test procedures.

Additional Traffic Signal Controller Test Cases			
ID	Title	Description	
dbCreat	eTransaction (DCT)		
TC0001	dbCreateTransaction	Verifies that the dbCreateTransaction transitions from all states to another properly, object values are buffered when in the transaction state, and a consistency check is performed.	
Intersect	ion Map (IM)		
TC0001	Intersection Map	Tests whether controller can produce the phase related parameters of a typical eight-phase, dual ring controller.	
Global C	Configuration (GloCon)		
TC0001	Retrieve Module Table	This procedure retrieves the module table, and allows the Tester to verify that the DUT reports the proper type of device, manufacturer, model, and version.	
TC0002	Global Set ID	This procedure ensures that a change to a static database object produces a change in globalSetIDParmeter.	
Retrieve	Log Data (RLD)		
TC0001	Setup Classes and Configure Events	This procedure checks whether objects are instantiated, sets up the eventClassTable and the eventLogConfigTable.	
TC0002	Check Class 1 Events	This procedure checks for proper functioning of the "onChange" eventLogConfig.Mode and whether the resulting log entries are produced.	
TC0003	Check Class 2 Events	This procedure checks for proper functioning of the "greaterThan" eventLogConfig.Mode and whether the resulting log entries are produced.	
TC0004	Check Class 3 Events	This procedure checks for proper functioning of the "smallerThan" eventLogConfig.Mode and whether the resulting log entries are produced.	
TC0005	Check Class 4 Events	This procedure checks for proper functioning of the "hysteresis" eventLogConfig.Mode and whether the resulting log entries are produced.	
TC0006	Check Class 5 Events	This procedure checks for proper functioning of the "periodic" eventLogConfig.Mode and whether the resulting log entries are produced.	
TC0007	Check Class 6 Events	This procedure checks for proper functioning of the "andedWith" eventLogConfig.Mode and whether the resulting log entries are produced.	

Table G-2. Additional Traffic Signal Controller Test Case Summary.

Additional Traffic Signal Controller Test Cases			
ID	Title	Description	
TC0008	Class Events Cleanup	This procedure clears the events in the logs and resets the eventClassDescriptions to null.	
System N	Map (SM)		
TC0001	System Map	The purpose of this test procedure is to demonstrate and verify the retrieval of status information that would typically be viewed in a system map display.	
Timebas	e Schedule of Events (TBE)		
TC0001	Setup ¹	This procedure is used to set up a scheduling configuration.	
TC0002	Trigger Scheduled Events	This procedure tests whether the schedule of events occurs.	
Databas	e Upload and Download (DUD)	·	
TC0001	Setup	This procedure verifies that the DUT supports the ascBlock objects, verifies that the administrator can change the administrator community name stored in the DUT, and properly affects operations.	
TC0002	Upload ascBlocks	This procedure verifies that a single block of all 21 types can be uploaded.	
TC0003	Download ascBlocks	This procedure verifies that a single block of all 21 types can be downloaded.	
TC0004	Upload Multiple ascBlocks	This procedure verifies that multiple blocks of all 21 types can be uploaded.	
TC0005	Download Multiple ascBlocks	This procedure verifies that multiple blocks of all 21 types can be downloaded.	
Time and	d Date (TAD)		
TC0001	Time and Date ¹	The following procedures check for proper operation in regard to setting and displaying time and date, whether time is expressed as a counter (sign bit is ignored and unit transitions from 0x7F FF FF FF to 0x80 00 00 00). These procedures also check for proper operation when daylight savings is not enabled and when it is. Both transitions (spring ahead and fall back) are checked. Finally, the procedure checks that time zone value has the appropriate effect on controller local time. Since the overflow condition, transitioning from 0xFF FF FF FF to 0x00 00 00 00 occurs in the next century, it is not checked.	

Table G-2. Additional Traffic Signal Controller Test Case Summary (continued).

¹ Scripts for both globalLocalTimeDifferential and controllStandardTimeZone are available.

TEST CASES

The details of the four-phase diamond sequencing and detector operations test cases follow. The basic format of the test cases comes from the template that appears in NTCIP 8007 – Testing and Conformity Assessment Documentation within NTCIP Standards Publications (6). As presented here the test case format has additional fields for clarity and version control.

ASC PRL Information

Test Case:	Title: ASC PRL Information	
TC001	Description: This procedure retrieves minimum project requirem	nents and
	maximum values, checks for whether the required	objects are
	implemented, and performs a sampling of the supp	ported values
	for P and C objects.	
	Constants:	
	Variables:	
	Pass/Fail The DUT shall pass every verification step include	d within the
	Criteria: Test Case in order to pass the Test Case.	-
Test Step	Test Procedure	Results
Number		
1. (Static	CONFIGURE a list <asc and="" max="" oids="" static=""> that identifies read-</asc>	
Values)	only objects that define either maximum values that affect the	
	indexes of tables or static variables that affect limits on other	
2.	CONFIGURE COMMUNITY NAME OUT = "administrator"	
3.	FOR ObjectName = each objectName in < ASC Max and Static	
4.	GET ObjectName	Pass/Fail
5.	Record the value on the PRL	
6.		
7. (Object	CONFIGURE a list of objectNames that must be supported	
Support)		
8.	FOR [objectivame = each objectivame in Supported objectivame List	
9.	FOR all possible [Instance values]	D/E-1
10.		Pass/Fall
	Note: This loop performs the equivalent of a MIP Walk but uses	
11	NEXT [instance value]	
11.	NEXT [nistance value]	
12. 13 (Range	CONFIGURE a list $< \Delta$ SC Test Values> that identifies instances of	
Support)	objects to test and a value in which to test the object with	
14	FOR [objectNameInstance] = each objectNameInstance in < CCTV	
17.	Test Values>	
15.	GET [objectNameInstance]	Pass/Fail
16.	RECORD RESPONSE VALUE in [currentValue]	
17.	FOR [testValue] = each objectNameValue in <asc test="" values=""></asc>	
18.	SET [objectNameInstance] = [testValue]	Pass/Fail
19.	Record ObjectName, TestValue, and errorStatus	
20.	NEXT [testValue]	

21.	SET [objectNameInstance] = [currentValue]	Pass/Fail
22.	NEXT ObjectName		
23.	RECORD responses on the assoc	iated NTCIP PRL and note any	-
	anomalies.		
	Test Case F	Results	
Tootod Dy:		Date	Pass/Fail
Tested by.		Tested	
Test Case Notes:	<notes></notes>		
Version History:	v1.00 04/05/06 Initial draft – RDF	2	
-	v1.01 07/05/06 Updated notes – RDR		
	v1.02 07/19/06 Implemented script and proofed – JJ		

Four-Phase Diamond Sequencing

Toot Coool	Titlo:	Sequencing	
		Tests conversion of all transitions from one state t	
10001	Description:	resis sequencing of all transitions from one state to	o another
	Constants	using venicle calls (not detector calls).	
	Constants:	None	
	variables:		
	Pass/Fail	The DUT shall pass every verification step included	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
1.	CONFIGURE the c	controller for:	
	1. 4-phase diamon	d operation with Phase 3 and 7 in sequence	
	2. Connection to T	S 2 Tester Box (BIU's enabled)	
	3. Vehicle and Ped	estrian Recalls are all off	
	4. Default data load	ded	
	Note: The setup for	r different manufacturer controllers will be	
	different. Reference	e should be made to a document containing such	
	information.	, i i i i i i i i i i i i i i i i i i i	
	Eagle Configuratio	n Notes:	
	Unit Data – Sta	rtup & Misc – Alt Sequence = 16	
	Unit Data – Por	t 1 Data = Enable T&F 1=4. DET 1-4.	
	and Malfunction	n Unit	
	Phase Data – Ir	nitialization & N.A. Response Phase 4 and 7 =	
	Change "Dark"	TO "Inactive"	
	onango bant		
	Naztec Configurati	on Notes [.]	
	Econolite Notes		
2.	FOR Phase = 1 TC	0 16	
3.	GET phaseMini	mumGreen.Phase, phasePassage.Phase, and	
	phaseMaximum	11.Phase	

4.	RECORD RESPONSE VALUE in [currentMinGrn.Phase],	
	[currentPassage.Phase] and [currentMax1.Phase]	
	Note: These values will be restored at the end of the test case	
5	SET phaseMinimumGreen Phase = 1 phasePassage Phase =	
5.	3 ± 10 and phase Maximum 1 Phase = 5	
6	NEXT Phase	
Sequence from 2+	5 with Calls on 2 and $3 = 3+5$ and $2+5$	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
3	DELAY 1 Second	
4	GET ringStatus 1, ringStatus 2	
5.	WEND	
•		
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 2 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
17.	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	No (or)No it for 0 (5 Ore on Doot	
	Note: Walt for 2+5 Green Rest.	
20.	Set HIL Detector input 3 = On	
<u> </u>	DELAY .2 Seconds	
<u> </u>	GET phaseStatusGroupPhaseOns. 1, phaseStatusGroupPhaseOns.2	
23.	while phasestatusGroupPhaseOhs. 1- 0x12 AND phaseStatusGroupPhaseOns 2- 0x00	
24	DELAV 1 Second	
27.	GET nhaseStatusGrounPhaseOns 1	
20.	nhaseStatusGroupPhaseOns 2	
26	WEND	
_0.		
	<i>Note:</i> Wait until a change from 2+5.	
27.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1=	Pass/Fail
	0x14 AND phaseStatusGroupPhaseOns.2 = 0x00	
	Note: When it does change, it should change to 3+5.	
28.	Set HITL Detector Input 3 = Off	
29.	DELAY .2 Seconds	
30.	Set HITL Detector Input 2 = On	
31.	DELAY .2 Seconds	
32.	IF phaseStatusGroupPhaseOns.1≠ 0x14 OR	

	phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	
	TerminationRestore	
	<i>Note:</i> If it does not go to 3+5 then restore original values and then	
	exit.	
33.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
34.	WHILE phaseStatusGroupPhaseOns.1 = 0x14 AND	
	phaseStatusGroupPhaseOns.2 = 0x00	
35.	DELAY 1 Second	
36.	GET phaseStatusGroupPhaseOns.1,	
07	phaseStatusGroupPhaseOns.2	
31.	WEND	
	Note: Wait until a change from 3+5	
.38	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns 1=	Pass/Fail
00.	0x12 AND phaseStatusGroupPhaseOns.2 = $0x00$	
	Note: When it does change, it should change to 2+5.	
39.	Set HITL Detector Input 2 = Off	
40.	DELAY .2 Seconds	
41.	IF phaseStatusGroupPhaseOns.1≠ 0x12 OR	
	phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO	
	IerminationRestore	
	Note: If it does not go to 2+5 then restore original values and then	
	Note. If it does not go to 2 ± 3 then restore original values and then avit	
Sequence from 2+	5 with Calls on 2 and $4 = 4+5$ and $2+5$	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
б. ¬	Set HITL Detector Input 2 = On	
<u> </u>	DELAY .2 Seconds	
<u>δ.</u>	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
Э.	WHILE phaseStatusGroupPhaseOns. $1 \neq 0$ X IZ AND phaseStatusGroupPhaseOns. $2 \neq 0$ X00	
10	DELAY 1 Second	
11	GFT phaseStatusGroupPhaseOns.1.	
• • •	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14.	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
17.	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Meit for 215 Green Best	
20	Note: Walt for 2+5 Green Rest.	
20.	Set HITL Delector input 4 = On	

21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1= 0x12 AND	
_	phaseStatusGroupPhaseOns.1= 0x00	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns 1	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
	Note: Wait until a change from 2+5.	
27.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns 1 =	Pass/Fail
	0x18 AND phaseStatusGroupPhaseOns.2 = 0x00	
	<i>Note:</i> When it does change, it should change to 4+5.	
28.	Set HITL Detector Input 4 = Off	
29	DELAY 2 Seconds	
30	Set HITL Detector Input 2 = On	
31	DELAY 2 Seconds	
32	IF phaseStatusGroupPhaseOns $1 \neq 0x18 \text{ OR}$	
02.	n phaseStatusGroupPhaseOns 2 \neq 0x00 THEN EXIT (Procedure)	
	Note: If it does not go to 4+5, then restore original values and exit	
33	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
34	WHILE nhaseStatusGroupPhaseOns $1 = 0x18$ AND	
54.	r_{112} nhaseStatusStroup haseOns 2 = 0x00	
35	DELAV 1 Second	
36	GET nhaseStatusGrounPhaseOns 1	
00.	nhaseStatusGrounPhaseOns 2	
37.	WEND	
37.	WEND	
37.	WEND Note: Wait until a change from 4+5.	
37.	WEND <i>Note:</i> Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠	Pass/Fail
37. 38.	WEND <i>Note:</i> Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00	Pass/Fail
37. 38.	WEND <i>Note:</i> Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00	Pass/Fail
37. 38.	WEND <i>Note:</i> Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 <i>Note:</i> When it does change, it should be to 2+5.	Pass/Fail
37. 38. 39.	WEND <i>Note:</i> Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 <i>Note:</i> When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off	Pass/Fail
37. 38. <u>39.</u> 40.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds	Pass/Fail
37. 38. <u>39.</u> 40. 41.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR	Pass/Fail
37. 38. <u>39.</u> 40. 41.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination	Pass/Fail
37. 38. <u>39.</u> 40. 41.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore	Pass/Fail
37. 38. <u>39.</u> 40. 41.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore	Pass/Fail
37. 38. <u>39.</u> 40. 41.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then	Pass/Fail
37. 38. <u>39.</u> 40. 41.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then exit.	Pass/Fail
37. 38. 39. 40. 41. Sequence from 2+	 WEND <i>Note:</i> Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 <i>Note:</i> When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore <i>Note:</i> If it does not go to 2+5, then restore original values and then exit. 5 with Calls on 2 and 6 = 5+9, 1+6, 1+13, and 2+5 	Pass/Fail
37. 38. 39. 40. 41. Sequence from 2+ 1.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then exit. 5 with Calls on 2 and 6 = 5+9, 1+6, 1+13, and 2+5 GET ringStatus.1, ringStatus.2	Pass/Fail
37. 38. 39. 40. 41. Sequence from 2+ 1. 2.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then exit. 5 with Calls on 2 and 6 = 5+9, 1+6, 1+13, and 2+5 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	Pass/Fail
37. 38. 39. 40. 41. Sequence from 2+ 1. 2. 3.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x0te: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then exit. 5 with Calls on 2 and 6 = 5+9, 1+6, 1+13, and 2+5 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	Pass/Fail
37. 38. 39. 40. 41. Sequence from 2+: 1. 2. 3. 4.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then exit. 5 with Calls on 2 and 6 = 5+9, 1+6, 1+13, and 2+5 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	Pass/Fail
37. 38. 39. 40. 41. Sequence from 2+ 1. 2. 3. 4. 5.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then exit. 5 with Calls on 2 and 6 = 5+9, 1+6, 1+13, and 2+5 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	Pass/Fail
37. 38. 39. 40. 41. Sequence from 2+ 1. 2. 3. 4. 5.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then exit. 5 with Calls on 2 and 6 = 5+9, 1+6, 1+13, and 2+5 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	Pass/Fail
37. 38. 39. 40. 41. Sequence from 2+ 1. 2. 3. 4. 5.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then exit. 5 with Calls on 2 and 6 = 5+9, 1+6, 1+13, and 2+5 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere.	Pass/Fail
37. 38. 39. 40. 41. Sequence from 2+ 1. 2. 3. 4. 5. 6.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then exit. 5 with Calls on 2 and 6 = 5+9, 1+6, 1+13, and 2+5 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 2 = On	Pass/Fail
37. 38. 39. 40. 41. Sequence from 2+ 1. 2. 3. 4. 5. 6. 7.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then exit. 5 with Calls on 2 and 6 = 5+9, 1+6, 1+13, and 2+5 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 2 = On DELAY .2 Seconds	Pass/Fail
37. 38. 39. 40. 41. Sequence from 2+ 1. 2. 3. 4. 5. 6. 7. 8.	WEND Note: Wait until a change from 4+5. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠ 0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00 Note: When it does change, it should be to 2+5. Set HITL Detector Input 2 = Off DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x12 OR phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO Termination Restore Note: If it does not go to 2+5, then restore original values and then exit. Swith Calls on 2 and 6 = 5+9, 1+6, 1+13, and 2+5 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 2 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	Pass/Fail

	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
17	DELAY 1 Second	
17.	GET ringStatus 1 ringStatus 2	
10.	WEND	
19.	WEND	
	Note: Wait for 2+5 Green Rest	
20	Set HITL Detector Input 6 - On	
20.	DELAX 2 Secondo	
<u>کا.</u>	DELAT .2 Seconds	
22.	GET phaseStatusGroupPhaseOns. 1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1= 0x12 AND	
~	phaseStatusGroupPhaseOns.1= 0x00	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
	Note: Wait until a change from 2+5.	
27.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x10 AND phaseStatusGroupPhaseOns.2 = 0x01	
	<i>Note:</i> When it does change, it should change to 5+9.	
28.	<i>Note:</i> When it does change, it should change to $5+9$. IF phaseStatusGroupPhaseOns.1 \neq 0x10 OR	
28.	<i>Note:</i> When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 \neq 0x10 OR phaseStatusGroupPhaseOns.2 \neq 0x01 THEN GOTO	
28.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore	
28.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore	
28.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then	
28.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit.	
28. 29.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
28. 29. 30.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND	
28. 29. 30.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01	
28. 29. 30. 31.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second	
28. 29. 30. 31. 32.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1,	
28. 29. 30. 31. 32.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2	
28. 29. 30. 31. 32. 33.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, wEND	
28. 29. 30. 31. 32. 33.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9.	
28. 29. 30. 31. 32. 33. 34.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
28. 29. 30. 31. 32. 33. 34.	Note: When it does change, it should change to 5+9.IF phaseStatusGroupPhaseOns.1 ≠ 0x10 ORphaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTOTerminationRestoreNote: If it does not go to 5+9 then restore original values and thenexit.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1= 0x10 ANDphaseStatusGroupPhaseOns.1= 0x01DELAY 1 SecondGET phaseStatusGroupPhaseOns.2WENDNote: Wait until a change from 5+9.VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x00	Pass/Fail
28. 29. 30. 31. 32. 33. 34.	Note: When it does change, it should change to 5+9.IF phaseStatusGroupPhaseOns.1 ≠ 0x10 ORphaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTOTerminationRestoreNote: If it does not go to 5+9 then restore original values and thenexit.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1= 0x10 ANDphaseStatusGroupPhaseOns.1= 0x01DELAY 1 SecondGET phaseStatusGroupPhaseOns.1,phaseStatusGroupPhaseOns.2WENDNote: Wait until a change from 5+9.VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x00	Pass/Fail
28. 29. 30. 31. 32. 33. 34.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x21 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+6.	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x21 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+6. Set HITL Detector Input 6 = Off	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35. 36	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x21 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+6. Set HITL Detector Input 6 = Off DELAY 2 Seconds	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35. 36. 37.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x21 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+6. Set HITL Detector Input 6 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x21 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+6. Set HITL Detector Input 6 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On DELAY .2 Seconds	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 30. 30. 30. 30. 30. 30. 30. 30	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x21 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+6. Set HITL Detector Input 6 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On DELAY .2 Seconds IF phaseStatusGroupPhaseOns 1 ≠ 0x21 OP	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x21 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+6. Set HITL Detector Input 6 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1 ≠ 0x21 OR phaseStatusGroupPhaseOns.1 ≠ 0x21 OR phaseStatusGroupPhaseOns.1 ≠ 0x21 OR	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x21 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+6. Set HITL Detector Input 6 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1 ≠ 0x21 OR phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	Pass/Fail

	Note: If it does not go to 1+6 then restore original values and then	
	exit.	
40.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
41.	WHILE phaseStatusGroupPhaseOns.1 = 0x21 AND	
	phaseStatusGroupPhaseOns.2 = 0x00	
42.	DELAY 1 Second	
43.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
44.	WEND	
	Note: Wait until a change from 1+6.	
45.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠	Pass/Fail
	0x01 AND phaseStatusGroupPhaseOns.1≠ 0x10	
	<i>Note:</i> When it does change, it should be to 1+13.	
46.	IF phaseStatusGroupPhaseOns.1≠ 0x01 OR	
	phaseStatusGroupPhaseOns.1≠ 0x10 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 1+13 then restore original values and then	
	exit.	
47.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
48.	WHILE phaseStatusGroupPhaseOns.1 = 0x01 AND	
	phaseStatusGroupPhaseOns.2 = 0x10	
49.	DELAY 1 Second	
50.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
51.	WEND	
	Note: Wait until a change from 1+13.	
52.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1≠	Pass/Fail
	0x12 AND phaseStatusGroupPhaseOns.1≠ 0x00	
	<i>Note:</i> When it does change, it should be to 2+5.	
53.	Set HITL Detector Input 2 = Off	
54.	DELAY .2 Seconds	
55.	IF phaseStatusGroupPhaseOns.1≠ 0x12 OR	
	phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO	
	TerminationRestore	
	<i>Note:</i> If it does not go to 2+5 then restore original values and then	
	exit.	
Sequence from 2+	5 with Calls on 2 and 7 = 5+9, 1+7, 2+15, and 2+5	[
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 7 UXU3 AND ringStatus.2 7 UXU3	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	<i>Note:</i> Loop until controller rests in green somewhere.	
<u>б</u> . ¬	Set HILL Detector Input 2 = On	
<i>. . . .</i>	DELAY 2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	

	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14	DELAY 2 Seconds	
15	GET ringStatus 1 ringStatus 2	
10.	WHILE ringStatus 1 \pm 0x03 AND ringStatus 2 \pm 0x03	
17	DELAV 1 Second	
10	CET ringStatus 1 ringStatus 2	
10.		
19.	WEND	
	No to Milit for 0 to Octobe Deat	
	Note: Walt for 2+5 Green Rest.	
20.	Set HIIL Detector Input 7 = On	
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1= 0x12 AND	
	phaseStatusGroupPhaseOns.1= 0x00	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
	Note: Wait until a change from 2+5.	
27	VERIEV that RESPONSE VALUE phaseStatusGroupPhaseOns 1 =	Pass/Fail
	0x10 AND phaseStatusGroupPhaseOns 2 = 0x01	
	<i>Note:</i> When it does change, it should change to 5+9.	
28	<i>Note:</i> When it does change, it should change to $5+9$.	
28.	<i>Note:</i> When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 \neq 0x10 OR phaseStatusGroupPhaseOns 2 \neq 0x01 THEN GOTO	
28.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore	
28.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore	
28.	<i>Note:</i> When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 \neq 0x10 OR phaseStatusGroupPhaseOns.2 \neq 0x01 THEN GOTO TerminationRestore	
28.	<i>Note:</i> When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 \neq 0x10 OR phaseStatusGroupPhaseOns.2 \neq 0x01 THEN GOTO TerminationRestore <i>Note:</i> If it does not go to 5+9 then restore original values and then exit	
28.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
28. 29. 30	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1	
28. 29. 30.	Note: When it does change, it should change to 5+9.IF phaseStatusGroupPhaseOns.1 ≠ 0x10 ORphaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTOTerminationRestoreNote: If it does not go to 5+9 then restore original values and thenexit.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1= 0x10 ANDphaseStatusGroupPhaseOns.1= 0x01	
28. 29. 30.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01	
28. 29. 30. 31.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second	
28. 29. 30. 31. 32.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1,	
28. 29. 30. 31. 32.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.2,	
28. 29. 30. 31. 32. 33.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.2	
28. 29. 30. 31. 32. 33.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9.	
28. 29. 30. 31. 32. 33. 34.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
28. 29. 30. 31. 32. 33. 34.	Note: When it does change, it should change to 5+9.IF phaseStatusGroupPhaseOns.1 ≠ 0x10 ORphaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTOTerminationRestoreNote: If it does not go to 5+9 then restore original values and thenexit.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1= 0x10 ANDphaseStatusGroupPhaseOns.1= 0x01DELAY 1 SecondGET phaseStatusGroupPhaseOns.2WENDNote: Wait until a change from 5+9.VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x00	Pass/Fail
28. 29. 30. 31. 32. 33. 34.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x00 Mute with the statusGroupPhaseOns.2 = 0x00	Pass/Fail
28. 29. 30. 31. 32. 33. 34.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x41 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+7.	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35.	Note: When it does change, it should change to 5+9.IF phaseStatusGroupPhaseOns.1 ≠ 0x10 ORphaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTOTerminationRestoreNote: If it does not go to 5+9 then restore original values and thenexit.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1= 0x10 ANDphaseStatusGroupPhaseOns.1= 0x01DELAY 1 SecondGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WENDNote: Wait until a change from 5+9.VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x41 AND phaseStatusGroupPhaseOns.2 = 0x00Note: When it does change, it should change to 1+7.Set HITL Detector Input 7 = Off	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35. 36.	Note: When it does change, it should change to 5+9.IF phaseStatusGroupPhaseOns.1 ≠ 0x10 ORphaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTOTerminationRestoreNote: If it does not go to 5+9 then restore original values and thenexit.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1= 0x10 ANDphaseStatusGroupPhaseOns.1= 0x01DELAY 1 SecondGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1= 0x01DELAY 1 SecondGET phaseStatusGroupPhaseOns.2WENDNote: Wait until a change from 5+9.VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x41 AND phaseStatusGroupPhaseOns.2 = 0x00Note: When it does change, it should change to 1+7.Set HITL Detector Input 7 = OffDELAY 2 Seconds	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35. 36. 37.	Note: When it does change, it should change to 5+9.IF phaseStatusGroupPhaseOns.1 ≠ 0x10 ORphaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTOTerminationRestoreNote: If it does not go to 5+9 then restore original values and thenexit.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1= 0x10 ANDphaseStatusGroupPhaseOns.1= 0x01DELAY 1 SecondGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WENDNote: Wait until a change from 5+9.VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x41 AND phaseStatusGroupPhaseOns.2 = 0x00Note: When it does change, it should change to 1+7.Set HITL Detector Input 7 = OffDELAY .2 SecondsSet HITL Detector Input 2 = On	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38.	Note: When it does change, it should change to 5+9.IF phaseStatusGroupPhaseOns.1 ≠ 0x10 ORphaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTOTerminationRestoreNote: If it does not go to 5+9 then restore original values and thenexit.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1= 0x10 ANDphaseStatusGroupPhaseOns.1= 0x01DELAY 1 SecondGET phaseStatusGroupPhaseOns.1,phaseStatusGroupPhaseOns.2WENDNote: Wait until a change from 5+9.VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =0x41 AND phaseStatusGroupPhaseOns.2 = 0x00Note: When it does change, it should change to 1+7.Set HITL Detector Input 7 = OffDELAY .2 SecondsSet HITL Detector Input 2 = OnDELAY .2 Seconds	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39.	Note: When it does change, it should change to 5+9.IF phaseStatusGroupPhaseOns.1 ≠ 0x10 ORphaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTOTerminationRestoreNote: If it does not go to 5+9 then restore original values and thenexit.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1= 0x10 ANDphaseStatusGroupPhaseOns.1= 0x01DELAY 1 SecondGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WENDNote: Wait until a change from 5+9.VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x41 AND phaseStatusGroupPhaseOns.2 = 0x00Note: When it does change, it should change to 1+7.Set HITL Detector Input 7 = OffDELAY .2 SecondsSet HITL Detector Input 2 = OnDELAY .2 SecondsIF phaseStatusGroupPhaseOns.1 ≠ 0x41 OR	Pass/Fail
28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39.	Note: When it does change, it should change to 5+9. IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO TerminationRestore Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1= 0x10 AND phaseStatusGroupPhaseOns.1= 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x41 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On DELAY .2 Seconds IF phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	Pass/Fail

	Note: If it does not go to 1+7 then restore original values and then	
	exit.	
40.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
41.	WHILE phaseStatusGroupPhaseOns.1 = 0x41 AND	
	phaseStatusGroupPhaseOns.2 = 0x00	
42.	DELAY 1 Second	
43.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
44.	WEND	
	Note: Moit until a change from 117	
	VOIE: Wall until a change from 1+7.	
40.	VERIFY THAT RESPONSE VALUE PHASeStatusGroupPhaseOns. IF	Pass/Fall
	0x02 AND phasestatus group maseons. 17 0x40	
	Note: When it does change, it should be to 2+15	
46	IF phaseStatusGroupPhaseOns 1± 0x02 OR	
40.	phaseStatusGroupPhaseOns 1≠ 0x40 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 2+15 then restore original values and then	
	exit.	
47.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
48.	WHILE phaseStatusGroupPhaseOns.1 = 0x02 AND	
	phaseStatusGroupPhaseOns.2 = 0x40	
49.	DELAY 1 Second	
50.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
51.	WEND	
F 0	Note: Walt until a change from 2+15.	D / C - 11
52.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.17	Pass/Fall
	0x12 AND phasestatusGroupPhaseOns. 17 0x00	
	Note: When it does change, it should be to 2+5	
53	Set HITI Detector Input 2 = Off	
50. 54	DELAY 2 Seconds	
55	IF phaseStatusGroupPhaseOns 1≠ 0x12 OR	
00.	phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 2+5 then restore original values and then	
	exit.	
Sequence from 2+	5 with Calls on 2 and 8 = 5+9, 1+8, 2+16, and 2+5	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	No feel and set the sector line and a line and a sector in the sector is a sector being	
6	Note: Loop until controller rests in green somewhere.	
0. 7	Set $\Pi \cap L$ Detector input 2 = $O \Pi$	
/ . o	DELAT .2 3600108	
<u>δ</u> .	UET phaseStatusGroupPhaseOns. 1, phaseStatusGroupPhaseOns.2	
9.	WHILE PHASESIAUSGIOUPHASEONS. I ≠ UX12 AND	

	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14	DELAY 2 Seconds	
15	GET ringStatus 1 ringStatus 2	
10.	WHILE ringStatus 1 \pm 0x03 AND ringStatus 2 \pm 0x03	
10.	DELAV 1 Second	
17.	CET ringStatus 1 ringStatus 2	
10.		
19.	WEND	
	Noto: Moit for 215 Oroon Doot	
	Note. Wall for 2+5 Green Rest.	
20.	Set HILL Detector Input 8 = On	
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1= 0x12 AND	
	phaseStatusGroupPhaseOns.1= 0x00	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
	Note: Wait until a change from 2+5.	
27.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x10 AND phaseStatusGroupPhaseOns.2 = 0x01	
	Note: When it does change, it should change to 5+9.	
28.	IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO	
	To making the post of the second	
	IerminationRestore	
	TerminationRestore	
	Note: If it does not go to 5+9 then restore original values and then	
	Note: If it does not go to 5+9 then restore original values and then exit.	
29.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
<u> </u>	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND	
29. 30.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01	
29. 30. 31.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second	
29. 30. 31. 32	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns 1	
29. 30. 31. 32.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
29. 30. 31. 32.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
29. 30. 31. 32. 33.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9	
29. 30. 31. 32. 33.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIEX that RESPONSE VALUE phaseStatusGroupPhaseOns 1 =	Pass/Fail
29. 30. 31. 32. 33. 34.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x81 AND phaseStatusGroupPhaseOns 2 = 0x00	Pass/Fail
29. 30. 31. 32. 33. 34.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x81 AND phaseStatusGroupPhaseOns.2 = 0x00	Pass/Fail
29. 30. 31. 32. 33. 34.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x81 AND phaseStatusGroupPhaseOns.2 = 0x00	Pass/Fail
29. 30. 31. 32. 33. 34.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x81 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+8. Set HITL Detector Input 8 = Off	Pass/Fail
29. 30. 31. 32. 33. 34. 35. 26	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x81 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+8. Set HITL Detector Input 8 = Off	Pass/Fail
29. 30. 31. 32. 33. 34. 35. 36.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x81 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+8. Set HITL Detector Input 8 = Off DELAY .2 Seconds	Pass/Fail
29. 30. 31. 32. 33. 34. 35. 36. 37.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x81 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+8. Set HITL Detector Input 8 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On	Pass/Fail
29. 30. 31. 32. 33. 34. 35. 36. 37. 38.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x81 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+8. Set HITL Detector Input 8 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On DELAY .2 Seconds	Pass/Fail
29. 30. 31. 32. 33. 34. 34. 35. 36. 37. 38. 39.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x81 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+8. Set HITL Detector Input 8 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On DELAY .2 Seconds	Pass/Fail
29. 30. 31. 32. 33. 34. 34. 35. 36. 37. 38. 39. 40.	Note: If it does not go to 5+9 then restore original values and then exit. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 = 0x10 AND phaseStatusGroupPhaseOns.1 = 0x01 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until a change from 5+9. VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 = 0x81 AND phaseStatusGroupPhaseOns.2 = 0x00 Note: When it does change, it should change to 1+8. Set HITL Detector Input 8 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On DELAY .2 Seconds IF phaseStatusGroupPhaseOns.1 ≠ 0x81 OR	Pass/Fail

	Restore	
	Note: If it does not go to 1+8 then restore original values and then	
	exit.	
41.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
42.	WHILE phaseStatusGroupPhaseOns.1 = 0x81 AND	
	phaseStatusGroupPhaseOns.2 = 0x00	
43.	DELAY 1 Second	
44.	GET phaseStatusGroupPhaseOns.1,	
·	phaseStatusGroupPhaseOns.2	
45.	WEND	
	Nata: Mait until a abanga from 1+9	
	VEDIEV that DESDONISE VALUE phaseStatusGroupDhaseOne 1-	Dace/Eail
40.	VERIFT IIIdI RESFONSE VALUE pildseSidiusGioupFildseOlis. I- 0x02 AND phaseStatusGroupPhaseOns 1- 0x80	rass/raii
	0x02 AND phasestatusoroupi haseons. 1- 0x00	
	<i>Note:</i> When it does change, it should be to 2+16.	
47.	IF phaseStatusGroupPhaseOns.1≠ 0x02 OR	
	phaseStatusGroupPhaseOns.1≠ 0x80 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 2+16 then restore original values and then	
	exit.	
48.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
49.	WHILE phaseStatusGroupPhaseOns.1 = 0x02 AND	
	phaseStatusGroupPhaseOns.2 = 0x80	
50.	DELAY 1 Second	
51.	GET phaseStatusGroupPhaseOns. 1,	
5 2		
52.	WEND	
	Note: Wait until a change from 2+16.	
53.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1=	Pass/Fail
	0x12 AND phaseStatusGroupPhaseOns.1 = 0x00	
	<i>Note:</i> When it does change, it should be to 2+5.	
54.	Set HITL Detector Input 2 = Off	
55.	DELAY .2 Seconds	
56.	IF phaseStatusGroupPhaseOns.1≠ 0x12 OR	
	phaseStatusGroupPhaseOns.1≠ 0x00 THEN GOTO	
	I erminationRestore	
	Note: If it does not go to 2+5 then restore original values and then	
	Avit	
Sequence from 3+	5 with Calls on 3 and $4 = 4+5$ and $3+5$	
1.	GET ringStatus 1, ringStatus 2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	

9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11	GET phaseStatusGroupPhaseOns 1	
	nhaseStatusGrounPhaseOns 2	
12		
12.	Veter Wait until controller reaches 2+5	
40	Note, Walt until controller reaches 5+5.	
13.	Set HITL Detector input 3 = On	
14.	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Wait for 3+5 Green Rest.	
20.	Set HITL Detector Input 4 = On	
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1= 0x14 AND	
-	phaseStatusGroupPhaseOns.1= 0x00	
24	DFLAY 1 Second	
25	GET phaseStatusGroupPhaseOns 1	
20.	nhaseStatusGrounPhaseOns 2	
26		
20.	Note: Wait until a change from 3+5	
27	VEDIEV that DESDONSE VALUE phaseStatusCroupDhaseOns 1 -	Dass/Fail
21.	V = 100000000000000000000000000000000000	r ass/r all
	0x10 AND phasestatusoroup-haseons.2 - 0x00	
	Noto: When it does change, it should change to 4+5	
<u></u>	Set UIT. Detector input 4 = Off	
20.	DELAV 2 Secondo	
29.	DELAT .2 Seconds	
30.	Set HITL Detector input 3 = On	
31.	DELAY .2 Seconds	
32.	IF phaseStatusGroupPhaseOns.1 ≠ 0x18 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	
	TerminationRestore	
	<i>Note:</i> If it does not go to 4+5 then restore original values and then	
	exit.	
33.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
34.	WHILE phaseStatusGroupPhaseOns.1 = 0x18 AND	
	phaseStatusGroupPhaseOns.2 = 0x00	
35.	DELAY 1 Second	
36.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
37.	WEND	
	<i>Note:</i> Wait until a change from 4+5.	
38.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x14 AND phaseStatusGroupPhaseOns.2 = 0x00	
	Note: When it does change, it should change to 3+5.	
39.	Set HITL Detector Input 3 = Off	
40.	DELAY .2 Seconds	

41.	IF phaseStatusGroupPhaseOns.1 ≠ 0x14 OR phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO TerminationRestore	
	<i>Note:</i> If it does not go to 3+5 then restore original values and then exit.	
Sequence from 3+	5 with Calls on 3 and 6 = 6+11, 1+6, 1+13, and 3+5	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
3.	DELAY 1 Second	
4	GET ringStatus 1 ringStatus 2	
5	WEND	
0.		
	Note: Loop until controller rests in green somewhere	
6	Set HITL Detector Input 3 = On	
7	DELAY 2 Seconds	
<u> </u>	CET nhaseStatusCrounPhaseOns 1_nhaseStatusCrounPhaseOns 2	
0.	$MHII = phaseStatusGroupPhaseOns 1 \neq 0x14 AND$	
9.	while phase status Group Phase Ons. $1 \neq 0.00$	
10	DELAV 1 Second	
10.	DELAT I Secolia	
11.	GET pridseStatusGroupPridseOns.1,	
10	phaseStatusGroupPhaseOns.2	
1Z.	WEND	
	Note: wait until controller reaches 3+5.	
13.	Set HIL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
17.	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Wait for 3+5 Green Rest.	
20.	Set HITL Detector Input 6 = On	
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1= 0x14 AND	
	phaseStatusGroupPhaseOns.2= 0x00	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
	Note: Wait until a change from 3+5.	
27.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x20 AND phaseStatusGroupPhaseOns.2 = 0x04	
	Note: When it does change, it should change to 6+11	
28.	IF phaseStatusGroupPhaseOns.1 ≠ 0x20 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x04 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 6+11 then restore original values and then	
	exit.	
29.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	

30.	WHILE phaseStatusGroupPhaseOns.1= 0x20 AND	
21	DELAV 1 Second	
ວາ. ວາ	DELAT I Secolu	
32.	GET phaseStatusGroupPhaseOns.1,	
່າງ		
33.	WEND Nata Wait until a change from 6 111	
24		
34.	0x21 AND phaseStatusGroupPhaseOns.2 = 0x00	Pass/Fall
	<i>Note:</i> When it does change, it should change to 1+6.	
35.	Set HITL Detector Input 6 = Off	
36.	DELAY .2 Seconds	
37.	Set HITL Detector Input 3 = On	
38	DELAY 2 Seconds	
30	IF phaseStatusGroupPhaseOns $1 \neq 0x21 \text{ OR}$	
	phaseStatusGroupPhaseOns.2 \neq 0x00 THEN GOTO TerminationRestore	
	exit.	
40.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
41.	WHILE phaseStatusGroupPhaseOns.1= 0x21 AND	
	phaseStatusGroupPhaseOns.2= 0x00	
42.	DELAY 1 Second	
43.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
44.	WEND	
	Note: Wait until a change from 1+6.	
45.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
-	0x01 AND phaseStatusGroupPhaseOns. 2 = 0x10	
	<i>Note:</i> When it does change, it should change to 1+13.	
46	IF phaseStatusGroupPhaseOns $1 \neq 0x01 \text{ OR}$	
10.	phaseStatusGroupPhaseOns 2 \neq 0x10 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 1+13 then restore original values and then	
	exit	
<u>4</u> 7	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
<u>18</u>	WHILE phaseStatusGroupPhaseOns 1= 0x01 AND	
40.	nhaseStatusGrounPhaseOns 2= 0x10	
/0	DELAV 1 Second	
	CET phaseStatusCroupPhaseOne 1	
50.	obaseStatusGroupPhaseOns 2	
E1		
51.	VVEND Noto: Wait until a change from 1+13	
5 0	V_{CD} Valuation of the transformation of	
JZ.	VERIFY LINAL RESPONSE VALUE PHASESIALUSGIOUPFHASEONS. I -	Fass/Fall
	0x14 AND phasestatusGroupPhaseOns.2 = 0x00	
	Nata Mhan it daga ahanna it ahauld ahanna ta 215	
	Note, when it does change, it should change to 3+5.	
53.	Set HIL Detector input $3 = OT$	
<u> </u>	DELAY 2 Seconds	
55.	IF phaseStatusGroupPhaseOns.1 ≠ 0x14 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	

	TerminationRestore	
	Nates If it does not go to 215 then restore privingly glues and then	
	vote: If it does not go to 3+5 then restore original values and then	
Sequence from 3+	5 with Calls on 3 and $7 = 5+9$ 1+7 1+13 and 3+5	
1.	GET ringStatus.1. ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
<u></u> 3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Wait for 3+5 Green Rest.	
20.	Set HILL Detector Input 7 = On	
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1= 0x14 AND	
	DELAX 1 Second	
24.	DELAT I Second	
25.	GET phaseStatusGroupPhaseOns.1,	
26		
20.	Note: Wait until a change from 3+5	
27	VERIEV that RESPONSE VALUE nhaseStatusGrounPhaseOns $1 =$	Pass/Fail
21.	0x10 AND phaseStatusGroupPhaseOns 2 = $0x01$	1 833/1 81
	<i>Note:</i> When it does change, it should change to 5+9.	
28.	IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 5+9 then restore original values and then	
	exit.	
29.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
30.	WHILE phaseStatusGroupPhaseOns.1= 0x10 AND	
	phaseStatusGroupPhaseOns.2= 0x01	

31.	DELAY 1 Second	
32.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
33.	WEND	
	<i>Note:</i> Wait until a change from 5+9.	
34.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x41 AND phaseStatusGroupPhaseOns.2 = 0x00	
~=	Note: When it does change, it should change to 1+7.	
35.	Set HIL Detector Input / = Off	
36.	DELAY 2 Seconds	
37.	Set HILL Detector Input 3 = On	
38.	DELAY .2 Seconds	
39.	IF phaseStatusGroupPhaseOns.1 ≠ 0x41 OR	
	pnaseStatusGroupPnaseOns.2 ≠ 0x00 THEN GOTO	
	reminationRestore	
	Note: If it does not go to $1+7$ then restore original values and then	
	exit	
40	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
40.	WHILE phaseStatusGroupPhaseOns 1= 0x41 AND	
тт.	phaseStatusGroupPhaseOns.2= 0x00	
42.	DELAY 1 Second	
43.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
44.	WEND	
	Note: Wait until a change from 1+7.	
45.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x01 AND phaseStatusGroupPhaseOns.2 = 0x10	
-	<i>Note:</i> When it does change, it should change to 1+13.	
46.	IF phaseStatusGroupPhaseOns.1 ≠ 0x01 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x10 THEN GOTO	
	reminationRestore	
	Note: If it does not go to $1+13$ then restore original values and then	
	avit	
<u>Δ</u> 7	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
48	WHILE phaseStatusGroupPhaseOns 1= 0x01 AND	
10.	phaseStatusGroupPhaseOns.2= 0x10	
49.	DELAY 1 Second	
50.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
51.	WEND	
	Note: Wait until a change from 1+13.	
52.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x14 AND phaseStatusGroupPhaseOns.2 = 0x00	
-	<i>Note:</i> When it does change, it should change to 3+5.	
53.	Set HIL Detector Input 3 = Off	
54.	DELAY .2 Seconds	
55.	IF phaseStatusGroupPhaseOns.1 \neq 0x14 OR	
	pnaseStatusGroupPnaseOns.2 ≠ 0x00 THEN GOTO	
	reminationRestore	

	Note: If it does not go to 3+5 then restore original values and then		
	exit.		
Sequence from 3+5 with Calls on 3 and 8 = 5+9, 1+8, 1+13, and 3+5			
1.	GET ringStatus.1, ringStatus.2		
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03		
3.	DELAY 1 Second		
4.	GET ringStatus.1, ringStatus.2		
5.	WEND		
	Note: Loop until controller rests in green somewhere.		
6.	Set HITL Detector Input 3 = On		
7.	DELAY .2 Seconds		
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2		
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND		
	phaseStatusGroupPhaseOns.2 ≠ 0X00		
10.	DELAY 1 Second		
11.	GET phaseStatusGroupPhaseOns.1,		
	phaseStatusGroupPhaseOns.2		
12.	WEND		
	Note: Wait until controller reaches 3+5.		
13.	Set HITL Detector Input 3 = Off		
14.	DELAY .2 Seconds		
15.	GET ringStatus.1, ringStatus.2		
16.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03		
17.	DELAY 1 Second		
18.	GET ringStatus.1, ringStatus.2		
19.	WEND		
	Note: Wait for 3+5 Green Rest.		
20.	Set HITL Detector Input 8 = On		
21.	DELAY .2 Seconds		
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2		
23.	WHILE phaseStatusGroupPhaseOns.1= 0x14 AND		
-	phaseStatusGroupPhaseOns.2= 0x00		
24.	DELAY 1 Second		
25.	GET phaseStatusGroupPhaseOns.1.		
	phaseStatusGroupPhaseOns.2		
26.	WEND		
	Note: Wait until a change from 3+5.		
27.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail	
	0x10 AND phaseStatusGroupPhaseOns.2 = 0x01		
	<i>Note:</i> When it does change, it should change to 5+9.		
28.	IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR		
-	phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO		
	TerminationRestore		
	<i>Note:</i> If it does not go to 5+9 then restore original values and then		
	exit.		
29.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2		
30.	WHILE phaseStatusGroupPhaseOns.1= 0x10 AND		
-	phaseStatusGroupPhaseOns.2= 0x01		
31.	DELAY 1 Second		
32.	GET phaseStatusGroupPhaseOns.1,		
	phaseStatusGroupPhaseOns.2		
-----	---	-----------	
33.	WEND		
	Note: Wait until a change from 5+9.		
34.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail	
	0x81 AND phaseStatusGroupPhaseOns.2 = 0x00		
	Note: When it does change, it should change to 1+8.		
35.	Set HITL Detector Input 8 = Off		
36.	DELAY .2 Seconds		
37.	Set HITL Detector Input 3 = On		
38.	DELAY .2 Seconds		
39.	IF phaseStatusGroupPhaseOns.1 ≠ 0x81 OR		
	phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO		
	TerminationRestore		
	Note: If it does not go to 1+8 then restore original values and then		
	exit.		
40.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2		
41.	WHILE phaseStatusGroupPhaseOns.1= 0x81 AND		
	phaseStatusGroupPhaseOns.2= 0x00		
42.	DELAY 1 Second		
43.	GET phaseStatusGroupPhaseOns.1,		
	phaseStatusGroupPhaseOns.2		
44.	WEND		
	Note: Wait until a change from 1+8.		
45.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail	
	0x01 AND phaseStatusGroupPhaseOns.2 = 0x10		
	Note: When it does change, it should change to 1+13.		
46.	IF phaseStatusGroupPhaseOns.1 ≠ 0x01 OR		
	phaseStatusGroupPhaseOns.2 ≠ 0x10 THEN GOTO		
	TerminationRestore		
	Note: If it does not go to 1+13 then restore original values and then		
	exit.		
47.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2		
48.	WHILE phaseStatusGroupPhaseOns.1= 0x01 AND		
	phaseStatusGroupPhaseOns.2= 0x10		
49.	DELAY 1 Second		
50.	GET phaseStatusGroupPhaseOns.1,		
	phaseStatusGroupPhaseOns.2		
51.	WEND		
	Note: Wait until a change from 1+13.		
52.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail	
	0x14 AND phaseStatusGroupPhaseOns.2 = 0x00		
	<i>Note:</i> When it does change, it should change to 3+5.		
53.	Set HITL Detector Input 3 = Off		
54.	DELAY .2 Seconds		
55.	IF phaseStatusGroupPhaseOns.1 ≠ 0x14 OR		
	phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO		
	TerminationRestore		
	<i>Note:</i> If it does not go to 3+5 then restore original values and then		
	exit.		

Sequence from 4+	5 with Calls on 4 and 6 = 6+12, 1+6, 1+13, and 4+5	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5	WEND	
0.		
	Note: Loop until controller rests in green somewhere	
6	Set HITL Detector Input 4 = On	
7	DELAV 2 Seconds	
<u> </u>	GET nhaseStatusGrounPhaseOns 1 nhaseStatusGrounPhaseOns 2	
<u> </u>	WHILE phaseStatusGroupPhaseOns $1 \neq 0x18$ AND	
5.	nhaseStatusGroupPhaseOns 2 + 0x00	
10	DELAV 1 Second	
10.	DELAY I Second	
TI.	GET phaseStatusGroupPhaseOns.1,	
	pnaseStatusGroupPnaseOns.2	
12.		
	/vote: vvalt until controller reaches 4+5.	
13.	Set HILL Detector Input 4 = Off	
	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
17.	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Wait for 4+5 Green Rest.	
20.	Set HITL Detector Input 6 = On	
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1= 0x18 AND	
	phaseStatusGroupPhaseOns.2= 0x00	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1.	
— -	phaseStatusGroupPhaseOns.2	
26.	WEND	
	Note: Wait until a change from 4+5.	
27	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
<u> </u>	0x20 AND phaseStatusGroupPhaseOns.2 = $0x08$	1 400/1 4
	Note: When it does change, it should change to 6+12.	
28	IF nhaseStatusGrounPhaseOns $1 \neq 0x20$ OR	
20.	nhaseStatusGrounPhaseOns 2 \neq 0x08 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 6+12 then restore original values and then	
	exit	
29	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
20.	WHILE phaseStatusGroupPhaseOne 1= 0x20 AND	
50.	nhaseStatusGroupPhaseOns 1= 0x08	
21	DELAV 1 Second	
20	CET phono Status Croup Phono One 1	
32.	GET pridseolalusGroupPridseolis.1,	
<u></u>		
33.	WEND	

	Note: Wait until a change from 6+12.	
34.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
_	0x21 AND phaseStatusGroupPhaseOns.2 = 0x00	
	<i>Note:</i> When it does change, it should change to 1+6.	
35.	Set HITL Detector Input 6 = Off	
36	DELAY 2 Seconds	
37	Set HITL Detector Input $4 = On$	
	DELAV 2 Seconds	
30.	DELAT .2 Occollus IE phaseStatusCroupPhaseOps $1 \neq 0x21 \text{ OP}$	
59.	$\frac{1}{2} = \frac{1}{2} + \frac{1}$	
	Termination Postero	
	TerminationRestore	
	Note: If it does not as to 1 if then restore original values and then	
40	exil.	
40.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
41.	WHILE phaseStatusGroupPhaseOns.1= 0x21 AND	
	phaseStatusGroupPhaseOns.2= 0x00	
42.	DELAY 1 Second	
43.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
44.	WEND	
	Note: Wait until a change from 1+6.	
45.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x01 AND phaseStatusGroupPhaseOns.2 = 0x10	
	Note: When it does change, it should change to 1+13.	
46.	IF phaseStatusGroupPhaseOns.1 ≠ 0x01 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x10 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 1+13 then restore original values and then	
	exit.	
47.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
48.	WHILE phaseStatusGroupPhaseOns.1= 0x01 AND	
	pnaseStatusGroupPnaseOns.2= 0x10	
49.	DELAY 1 Second	
50.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
51.	WEND	
	Note: Wait until a change from 1+13.	
52.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x18 AND phaseStatusGroupPhaseOns.2 = 0x00	
	Note: When it does change, it should change to 4+5.	
53.	Set HILL Detector Input 4 = Off	
54.	DELAY 2 Seconds	
55.	IF phaseStatusGroupPhaseOns.1 ≠ 0x18 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	
	IerminationRestore	
	<i>Note:</i> If it does not go to 4+5 then restore original values and then	
	exit.	
Sequence from 4+	5 with Calls on 4 and 7 = 5+9, 1+7, 1+13, and 4+5	
1.	GET ringStatus.1, ringStatus.2	

2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1. ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 4 = On	
7	DELAY 2 Seconds	
8	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
9. 9	WHILE phaseStatusGroupPhaseOns 1 \pm 0v18 AND	
0.	nhaseStatusGroupPhaseOns $2 \neq 0x00$	
10	DELAY 1 Second	
10.	GET nhaseStatusGrounPhaseOns 1	
11.	nhaseStatusGroupPhaseOns 2	
12		
12.	Note: Wait until controller reaches 4+5	
13	Set HITL Detector Input $4 = Off$	
11	DELAX 2 Seconda	
14.	CET ringStatue 1 ringStatue 2	
10.	GET IIIIgStatus T, IIIIgStatus Z	
10.	VHILE Ingstatus. 1 7 0x03 AND Ingstatus.2 7 0x03	
17.	DELAY 1 Second	
18.		
19.	WEND	
	Note Math from ALE Operation Depart	
	Note: Walt for 4+5 Green Rest.	
20.	Set HILL Detector Input / = On	
21.	DELAY 2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1= 0x18 AND	
	phaseStatusGroupPhaseOns.2= 0x00	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
	Note: Wait until a change from 4+5.	
27.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x10 AND phaseStatusGroupPhaseOns.2 = 0x01	
	Note: When it does change, it should change to 5+9.	
28.	IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO	
	TerminationRestore	
	Make If it does not up to FLO then notions evicingly volves and then	
	Note: If it does not go to 5+9 then restore original values and then	
	exit.	
29.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
30.	WHILE phaseStatusGroupPhaseOns.1= 0x10 AND	
	phaseStatusGroupPhaseOns.2= 0x01	
31.	DELAY 1 Second	
32.	GET phaseStatusGroupPhaseOns.1,	
	pnaseStatusGroupPnaseOns.2	
33.		
	<i>Note:</i> wait until a change from 5+9.	
34.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail

	0x41 AND phaseStatusGroupPhaseOns.2 = 0x00	
	Note: When it does change, it should change to 1+7	
25	Set HITL Detector Input $7 = \Omega ff$	
36 36	DELAY 2 Seconds	
37	Set HITL Detector Input $4 - \Omega n$	
20 20	DELAX 2 Seconde	
<u> </u>	IE phaseStatusCroupDhaseOps 1 + 0v41 OP	
	$\frac{1}{2} = \frac{1}{2} + \frac{1}$	
	TerminationRestore	
	Note: If it does not go to 1+7 then restore original values and then	
	exit	
40.	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
41	WHILE phaseStatusGroupPhaseOns 1= 0x41 AND	
	phaseStatusGroupPhaseOns 2= 0x00	
42	DELAY 1 Second	
43.	GET phaseStatusGroupPhaseOns 1.	
	phaseStatusGroupPhaseOns.2	
44.	WEND	
	<i>Note:</i> Wait until a change from 1+7.	
45.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x01 AND phaseStatusGroupPhaseOns.2 = 0x10	
	Note: When it does change, it should change to 1+13.	
46.	IF phaseStatusGroupPhaseOns.1 ≠ 0x01 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x10 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 1+13 then restore original values and then	
	exit.	
47.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
48.	WHILE phaseStatusGroupPhaseOns.1= 0x01 AND	
	phaseStatusGroupPhaseOns.2= 0x10	
49.	DELAY 1 Second	
50.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
51.		
	Note: Walt until a change from 1+13.	D / E - 11
52.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fall
	0x18 AND phaseStatusGroupPhaseOns.2 = 0x00	
	Nata: When it does alongo, it should alongo to 4+5	
52	Note: When it does change, it should change to $4+5$.	
53. 54	DELAV 2 Seconds	
55	DELAT 2 Seconds	
55.	The phase Status Group Phase Ons $2 \neq 0x00$ THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 4+5 then restore original values and then	
	exit.	
Sequence from 4+	5 with Calls on 4 and 8 = 5+9, 1+8, 1+13, and 4+5	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
3.	DELAY 1 Second	

4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
-		
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY 2 Seconds	
8	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
0. 0	WHILE phaseStatusGroupPhaseOns $1 \neq 0x18$ AND	
5.	$haseStatusGroupPhaseOns 2 \neq 0x00$	
10	DELAV 1 Second	
10.	GET nhaseStatusCrounPhaseOns 1	
11.	obaseStatusGroupPhaseOns 2	
10		
12.	WEIND Noto: Wait until controller reaches 4+5	
10		
13.	Set HITL Detector input 4 = OII	
14.	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Wait for 4+5 Green Rest.	
20.	Set HITL Detector Input 8 = On	
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1= 0x18 AND	
	phaseStatusGroupPhaseOns.2= 0x00	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
	Note: Wait until a change from 4+5.	
27.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x10 AND phaseStatusGroupPhaseOns. 2 = 0x01	
	- production of the second	
	<i>Note:</i> When it does change, it should change to 5+9.	
28.	IF phaseStatusGroupPhaseOns.1 ≠ 0x10 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x01 THEN GOTO	
	TerminationRestore	
	<i>Note:</i> If it does not go to 5+9 then restore original values and then	
	exit.	
29.	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
30.	WHILE phaseStatusGroupPhaseOns 1= 0x10 AND	
	phaseStatusGroupPhaseOns.2= 0x01	
31	DELAY 1 Second	
32	GET phaseStatusGroupPhaseOns 1	
02.	phaseStatusGroupPhaseOns 2	
.3.3	WEND	
	Note: Wait until a change from 5+9	
34	VERIEV that RESPONSE VALUE phaseStatusGroupPhaseOne 1 =	Pass/Fail
	0x81 AND phaseStatusGroupPhaseOns 2 = $0x00$	1 400/1 411

	<i>Note:</i> When it does change, it should change to 1+8.	
35.	Set HITL Detector Input 8 = Off	
36.	DELAY .2 Seconds	
37.	Set HITL Detector Input 4 = On	
38.	DELAY .2 Seconds	
39.	IF phaseStatusGroupPhaseOns.1 ≠ 0x81 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	
	TerminationRestore	
	<i>Note:</i> If it does not go to 1+8 then restore original values and then	
	exit.	
40.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
41.	WHILE phaseStatusGroupPhaseOns.1= 0x81 AND	
	phaseStatusGroupPhaseOns.2= 0x00	
42.	DELAY 1 Second	
43.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
44.	WEND	
	Note: Wait until a change from 1+8.	
45.	VERIEY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x01 AND phaseStatusGroupPhaseOns.2 = 0x10	
	<i>Note:</i> When it does change, it should change to 1+13.	
46.	IF phaseStatusGroupPhaseOns.1 ≠ 0x01 OR	
_	phaseStatusGroupPhaseOns.2 ≠ 0x10 THEN GOTO	
	TerminationRestore	
	<i>Note:</i> If it does not go to 1+13 then restore original values and then	
	exit.	
47.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
48.	WHILE phaseStatusGroupPhaseOns.1= 0x01 AND	
	phaseStatusGroupPhaseOns.2= 0x10	
49.	DELAY 1 Second	
50.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
51.	WEND	
	<i>Note:</i> Wait until a change from 1+13.	
52.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x18 AND phaseStatusGroupPhaseOns.2 = 0x00	
	<i>Note:</i> When it does change, it should change to 4+5.	
53.	Set HITL Detector Input 4 = Off	
54.	DELAY .2 Seconds	
55.	IF phaseStatusGroupPhaseOns.1 ≠ 0x18 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 4+5 then restore original values and then	
	exit.	
Sequence from 1+	6 with Calls on 6 and 7 = 1+7 and 1+6	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	

~	Note: Loop until controller rests in green somewhere.	
6.	Set HILL Detector Input 6 = On	
<u> </u>	DELAY .2 Seconds	
<u> </u>	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
17.	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Wait for 1+6 Green Rest.	
20.	Set HITL Detector Input 7 = On	
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1= 0x21 AND	
	phaseStatusGroupPhaseOns.1= 0x00	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
07	Note: Walt until a change from 1+6.	
27.	0x41 AND phaseStatusGroupPhaseOns.2 = 0x00	Pass/Fall
	<i>Note:</i> When it does change, it should change to 1+7.	
28.	Set HITL Detector Input 7 = Off	
29.	DELAY .2 Seconds	
30.	Set HITL Detector Input 6 = On	
31.	DELAY .2 Seconds	
32.	IF phaseStatusGroupPhaseOns.1 ≠ 0x41 OR	
-	phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	
	TerminationRestore	
	Note: If it does not go to 1+7 then restore original values and then	
	exit.	
33.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
34.	WHILE phaseStatusGroupPhaseOns.1= 0x41 AND	
35	DELAY 1 Second	
26.	GET nhaseStatusGrounPhaseOne 1	
50.	phaseStatusGroupPhaseOns 2	
37	WEND	
	Note: Wait until a change from 1+7.	
38.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	· · · · ·	

	0x21 AND phaseStatusGroupPhaseOns.2 = 0x00	
	<i>Note:</i> When it does change, it should change to 1+6.	
39.	Set HIL Detector Input 6 = Off	
40.	DELAY .2 Seconds	
41.	IF phaseStatusGroupPhaseOns.1 ≠ 0x21 OR	
	phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	
	TerminationRestore	
	Note if it does not go to 1.10 then restore evisional values and then	
	vote. In it does not go to 1+6 then restore original values and then	
Sequence from 1+	6 with Calls on 6 and 8 - 1+8 and 1+6	ļ
	GET ringStatus 1 ringStatus 2	
1. 	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
2.	DELAY 1 Second	
<u> </u>	GET ringStatus 1 ringStatus 2	
<u>т.</u> 5	WEND	
0.		
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY 2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
17.	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	<i>Note:</i> Walt for 1+6 Green Rest.	
20.		
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	while phaseStatusGroupPhaseOns. I= 0x21 AND	
	DELAY 1 Second	
24.	GET nhasoStatusGroupDhasoOns 1	
25.	nhaseStatusCroupPhaseOns 2	
26		
20.	Note: Wait until a change from 1+6	
27	VERIEV that RESPONSE VALUE phaseStatusGroupPhaseOns 1 =	Pass/Fail
£1.	0x81 AND phaseStatusGroupPhaseOns 2 = 0x00	
	<i>Note:</i> When it does change, it should change to 1+8.	
28.	Set HITL Detector Input 8 = Off	
29.	DELAY .2 Seconds	

31. DELAY .2 Seconds 32. IF phaseStatusGroupPhaseOns.1 ≠ 0x81 OR phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	
32. IF phaseStatusGroupPhaseOns.1 ≠ 0x81 OR phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	
phaseStatusGroupPhaseOns.2 ≠ 0x00 THEN GOTO	
TerminationRestore	
Note: If it does not go to 1+8 then restore original values and then	
exit.	
33. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
34. WHILE phaseStatusGroupPhaseOns.1= 0x81 AND	
phaseStatusGroupPhaseOns.2= 0x00	
35. DELAY 1 Second	
36. GET phaseStatusGroupPhaseOns.1.	
phaseStatusGroupPhaseOns.2	
37. WEND	
Note: Wait until a change from 1+8.	
38 VERIEV that RESPONSE VALUE phaseStatusGroupPhaseOns 1 = Pa	ass/Fail
$0x^{21}$ AND phaseStatusGroupPhaseOns 2 = $0x^{00}$	
Note: When it does change, it should change to 1+6	
39 Set HITL Detector Input 6 = Off	
40 DELAY 2 Seconds	
41 IF phaseStatusGroupPhaseOns $1 \neq 0x21$ OR	
phaseStatusGroupPhaseOns 2 \neq 0x00 THEN GOTO	
TerminationRestore	
Notes If it does not up to 4+0 them restand evidence and them	
/vore: If it does not do to 1+6 then restore original values and then	
exit.	
Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second	
Note: If it does not go to 1+6 then restore original values and then exit.Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+71.GET ringStatus.1, ringStatus.22.WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x033.DELAY 1 Second4.GET ringStatus.1, ringStatus.2	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere.	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On	
Note: In t does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On 7. DELAY .2 Seconds	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
Note: In t does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
Note: In t does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 0x41 AND	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second	
Note: In it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 10. 10. DELAY 1 Second	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 10. 10. DELAY 1 Second 11. GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2	
Nore: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 10. 11. GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2 12. WEND	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 10. 11. GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2 12. WEND Note: Wait until controller reaches 1+7.	
Note: If it does not go to 1+6 then restore original values and then exit.Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+71.GET ringStatus.1, ringStatus.22.WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x033.DELAY 1 Second4.GET ringStatus.1, ringStatus.25.WENDNote: Loop until controller rests in green somewhere.6.Set HITL Detector Input 7 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 \neq 0x41 ANDphaseStatusGroupPhaseOns.2 \neq 0x0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WENDNote: Wait until controller reaches 1+7.13.Set HITL Detector Input 7 = Off	
Note: If it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 10. 11. GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2 12. WEND Note: Wait until controller reaches 1+7. 13. Set HITL Detector Input 7 = Off 14. DELAY 2 Seconds	
Note: In it does not go to 1+6 then restore original values and then exit. Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+7 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests in green somewhere. 6. Set HITL Detector Input 7 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 10. 11. GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2 12. WEND Note: Wait until controller reaches 1+7. 13. Set HITL Detector Input 7 = Off 14. DELAY .2 Seconds 15. GET ringStatus 1, ringStatus 2	
Note: In the does not go to 1+6 then restore original values and then exit.Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+71.GET ringStatus.1, ringStatus.22.WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x033.DELAY 1 Second4.GET ringStatus.1, ringStatus.25.WENDNote: Loop until controller rests in green somewhere.6.Set HITL Detector Input 7 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 \neq 0x41 AND phaseStatusGroupPhaseOns.2 \neq 0x0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 1+7.13.Set HITL Detector Input 7 = Off14.DELAY .2 Seconds15.GET ringStatus.1, ringStatus.216.WHILE fringStatus.1, fringStatus.2	
Note: In it does not go to 1+6 then restore original values and then exit.Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+71.GET ringStatus.1, ringStatus.22.WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x033.DELAY 1 Second4.GET ringStatus.1, ringStatus.25.WENDNote: Loop until controller rests in green somewhere.6.Set HITL Detector Input 7 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 \neq 0x41 AND phaseStatusGroupPhaseOns.2 \neq 0x0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 1+7.13.Set HITL Detector Input 7 = Off14.DELAY .2 Seconds15.GET ringStatus.1, ringStatus.216.WHILE ringStatus.1, \neq 0x03 AND ringStatus.2 \neq 0x03	
Note: If it does not go to 1+6 then restore original values and then exit.Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+71.GET ringStatus.1, ringStatus.22.WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x033.DELAY 1 Second4.GET ringStatus.1, ringStatus.25.WENDNote: Loop until controller rests in green somewhere.6.Set HITL Detector Input 7 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 \neq 0x0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.2 \neq 0x0012.WENDNote: Wait until controller reaches 1+7.13.Set HITL Detector Input 7 = Off14.DELAY .2 Seconds15.GET ringStatus.1, ringStatus.216.WHILE ringStatus.1, ringStatus.217.DELAY 1 Second18.GET ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
Note: If it does not go to 1+6 then restore original values and then exit.Sequence from 1+7 with Calls on 7 and 8 = 1+8 and 1+71.GET ringStatus.1, ringStatus.22.WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x033.DELAY 1 Second4.GET ringStatus.1, ringStatus.25.WENDNote: Loop until controller rests in green somewhere.6.Set HITL Detector Input 7 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 \neq 0x41 AND phaseStatusGroupPhaseOns.2 \neq 0x0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 1+7.13.Set HITL Detector Input 7 = Off14.DELAY .2 Seconds15.GET ringStatus.1, ringStatus.216.WHILE ringStatus.1, ringStatus.217.DELAY .2 Seconds18.GET ringStatus.1, ringStatus.2	

	Note: Wait for 1+7 Green Rest.	
20.	Set HITL Detector Input 8 = On	
21	DELAY 2 Seconds	
27	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
22. 02	WHII = phaseStatusCroup PhaseOns.1, phaseStatusCroup HaseOns.2	
23.	while phaseStatusGroupPhaseOns. 1- 0x41 AND	
0.4		
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
	Note: Wait until a change from 1+ 7.	
27.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 =	Pass/Fail
	0x81 AND phaseStatusGroupPhaseOns.2 = 0x00	
	<i>Note:</i> When it does change, it should change to 1+8.	
28.	Set HITL Detector Input 8 = Off	
29.	DELAY 2 Seconds	
30	Set HITL Detector Input $7 = On$	
31	DELAV 2 Seconds	
<u>ວາ</u>	PELAT .2 Octomos	
32.	n phaseStatusGroupPhaseOns. 1 7 0x01 OR	
	Termination Destars	
	reminationRestore	
	Nate, If it does not go to 1.0 then restore original values and then	
	<i>ivote:</i> If it does not go to 1+8 then restore original values and then	
	exit.	
33.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
34.	WHILE phaseStatusGroupPhaseOns.1= 0x81 AND	
	phaseStatusGroupPhaseOns.2= 0x00	
35.	DELAY 1 Second	
36.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
37.	WEND	
	Note: Wait until a change from 1+ 8.	
38	VERIEY that RESPONSE VALUE phaseStatusGroupPhaseOns 1 =	Pass/Fail
00.	0x41 AND phaseStatusGroupPhaseOns 2 = $0x00$	
	Note: When it does change, it should change to 1+7	
30	Set HITL Detector Input 7 - Off	
40	DELAX = 2 Seconds	
40.	DELAT .2 Securius	
41.	IF phaseStatusGroupPhaseOns. I ≠ 0x41 OR	
	pnaseStatusGroupPnaseOns.2 ≠ 0x00 THEN GOTO	
	IerminationRestore	
	<i>Note:</i> If it does not go to 1+7 then restore original values and then	
	exit.	
Termination Restor		1
1. (-	FUR Phase = 1 10 16	
(Termination		
Restore)		
2.	SET phaseMinimumGreen.Phase = [currentMinGrn.Phase],	
	phasePassage.Phase = [currentPassage.Phase], and	
	phaseMaximum1.Phase = [currentMax1.Phase]	
3.	NEXT Phase	
4.	Set HITL Detector Input 2, 3, 4, 6, 7, and 8 = Off	
5.	DELAY .2 Seconds	

Test Case Results			
Tested By:		Date Tested	Pass/Fail
Test Case Notes:	<notes></notes>		
Version History:	v1.00 04/07/06 Initial draft – RD v1.01 07/05/06 Updated notes – v1.02 07/24/06 Implemented scr	R · RDR ·ipt and proofed – JJ	

Four-Phase Diamond Detector Operations

The test cases check the operation of detector inputs 1 through 18 when a traffic signal controller is configured for four-phase diamond operation. The test cases are defined upon the Detector Operation Requirements appearing on page 18 of TxDOT DMS-11170, Fully Actuated, Solid-State Traffic Signal Controller Assembly – dated August 2004 (4).

Test Case:	Title:	Detector 1 Operations	
TC001	Description: Verifies the operation of Detector 1 to call Phase 6 if overlap A		
		is not green and there is no call on phase 7 or 8, and extends	
		intervals 2516B, 2517B, 2518B, 4517B, 4518B, 15	17B, and
		3518B.	
	Constants:		
	Variables:		
	Pass/Fail	The DUT shall pass every verification step included	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detector	r Operations Setup – TC019 if not already done	
	SO.		
Detector 1 No Call	on Phase 6 when Ov	verlap A is Green	
1.	GET ringStatus.1, ri	ngStatus.2	
2.	WHILE ringStatus.1	\neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	d	
4.	GET ringStatus.1	1, ringStatus.2	
5.	WEND		
	Note: Loop until con	troller rests somewhere.	
6.	Set HITL Detector Ir	nput 6 = On	
7.	DELAY .2 Seconds		
8.	GET phaseStatusGr	roupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatus	GroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupP	PhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	d	
11.	GET phaseStatu	sGroupPhaseOns.1,	
	phaseStatusGrou	upPhaseOns.2	
12.	WEND		
	Note: Wait until cont	troller reaches 1+6.	
13.	Set HITL Detector Ir	nput 6 = Off	
14.	DELAY .2 Seconds		
15.	GET ringStatus.1, ri	ngStatus.2	
16.	WHILE ringStatus.1	\neq 0x03 AND ringStatus.2 \neq 0x03	
17.	DELAY 1 Second	d	
18.	GET ringStatus.1	1, ringStatus.2	

Detector 1 Operations

19.	WEND	
	Nata Mait fan 110 Oraan Daat	
20	Note: Walt for 1+6 Green Rest.	
20.	GET overlapStatusGroupGreens. I	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap $A = 1+2+X$.	
21.	IF RESPONSE ERROR = noError THEN	
22.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND 0x01 = 0x01	
	<i>Note:</i> Verifies that Overlap A = Green.	
23.	ENDIF	
24.	Set HITL Detector Input 1 = On	
25.	DELAY3 Seconds (3 full seconds)	
26.	GET phaseStatusGroupVehCalls.1	
27.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1	Pass/Fail
	AND $0x20 = 0x00$	
	Noto: Ensure that Phase 6 is not registering a Vehicle Call	
28	Set HITI. Detector Input 1 = Off	
20.	DELAY 2 Seconds	
Detector 1 No Call	on Phase 6 when Overlap A is not Green and a Call on Phase 7	
1.	GET ringStatus.1. ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0X18 AND	
10	DELAX 1 Second	
10.	CET phaseStatusGroupPhaseOps 1	
11.	nhaseStatusGrounPhaseOns 2	
12	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap $A = 1+2+X$.	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND $0x01 = 0x00$	
	N_{ato} Varifies that Overlap A = NOT Organ	
10		
10.	Set HITL Detector Input $4 = \Omega n$	
20	DELAY 2 Seconds	
<u> </u>		

21.	Set HITL Detector Input 7 = On	
22.	DELAY 2 Seconds (2 full seconds)	
23.	Set HITL Detector Input 1 = On	
24.	DELAY 2 Seconds (2 full seconds)	
25.	GET phaseStatusGroupVehCalls.1	
26	VERIEV that RESPONSE VALUE phaseStatusGroupVehCalls 1	Pass/Fail
20.		1 400/1 411
	Note: Ensure that Phase 6 is not registering a Vehicle Call	
27	Set HITL Detector Input 1 = Ω ff 7 = Ω ff and 4 = Ω ff	
28	DELAY 2 Seconds	
Detector 1 No Call	on Phase 6 when Overlan A is not Green and a Call on Phase 8	L
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 \pm 0x03 AND ringStatus 2 \pm 0x03	
<u>ک</u> . ۲	DELAV 1 Second	
<u> </u>	GET ringStatus 1 ringStatus 2	
т. Б		
5.	WEND	
	Note: Loop until controller rests somewhere	
6	Set HITL Detector Input 4 - On	
7	DELAX 2 Seconde	
· · · · · · · · · · · · · · · · · · ·	DELAT .2 Seconds	
0. 0	GET phaseStatusGroupPhaseOns. 1, phaseStatusGroupPhaseOns.2	
9.	while phaseStatusGroupPhaseOns. 1 7 0x18 AND	
10		
10.	DELAY I Second	
11.	GET phaseStatusGroupPhaseOns. 1,	
10		
12.	WEIND	
12	Note: Walt until controller reaches $4+5$.	
13.	DELAX 2 Seconda	
14.	CET everlepStatueCroupCroope 1	
15.	GET OverlapStatusGroupGreens. I	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlan $A = 1+2+X$	
16	IF $PESPONSE EPPOP = poError THEN$	
17	VEDIEV that DESDONSE VALUE overlanStatusGroupGroops 1	Dass/Eail
17.	$\Lambda = 0.00$	F 855/F 811
	Note: Verifies that Overlap $A = NOT$ Green	
18		
10.	Set HITL Detector Input $4 = On$	
20	DELAY 2 Seconds	
20.	Set HITI Detector Input 8 = On	
21.	DELAX 2 Seconds (2 full seconds)	
22.	Set HITL Detector Input 1 - On	
23.	DELAV 2 Seconde	
<u> </u>	CET phaseStatusGroupVahCalle 1	
20. 06	VEDIEV that DESDONSE VALUE phaseStatusCroupVahCalls 1	
20.		Pass/Fall
	AND $UXZU = UXUU$	
	Noto: Ensure that Phase 6 is not registering a Vahiela Call	
	Note: Ensure that Filase 0 is not registering a vehicle Gall. Set HIT Detector input 1 = Off 9 = Off and 4 = Off	
27.	Set THE Detector input $T = OII, \delta = OII, and 4 = OII$	
28.	DELAT .2 SECONDS	

Detector 1 calls Ph	ase 6 when Overlap A is not Green and no Calls on Phase 7 or Pha	se 8
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
-		
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY 2 Seconds	
15.	GET overlapStatusGroupGreens.1	
	<i>Note:</i> overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap $A = 1+2+X$.	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND 0x01 = 0x00	
	<i>Note:</i> Verifies that Overlap A = NOT Green	
18.	ENDIF	
19.	Set HITL Detector Input 1 = On	
20.	DELAY 2 Seconds (2 full seconds)	
21.	GET phaseStatusGroupVehCalls.1	
22.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1	Pass/Fail
	AND 0x20 = 0x20	
	Note: Check for call on Phase 6.	
23.	Set HITL Detector Input 1 = Off	
24.	DELAY .2 Seconds	
Detector 1 extends	Phase 9 when Phase 9 is Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	

	phaseStatusGroupPhaseOns.2		
12.	WEND		
	Note: Wait until controller reaches 4+5.		
13.	Set HITL Detector Input 4 = Off		
14.	DELAY .2 Seconds		
15.	Set HITL Detector Input 7 = On		
16.	DELAY .2 Seconds		
17.	Set HITL Detector Input 1 = On		
18.	DELAY .2 Seconds		
19.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2		
20.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x10 AND		
	phaseStatusGroupPhaseOns.2 ≠ 0X01		
21.	DELAY 1 Second		
22.	GET phaseStatusGroupPhaseOns.1,		
	phaseStatusGroupPhaseOns.2		
23.	WEND		
	Note: Wait until controller reaches 5+9.		
24.	GET ringStatus.1		
25.	WHILE ringStatus.1 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)		
26.	DELAY 1 Second		
27.	GET ringStatus.1		
28.	WEND		
	Note: Wait for Max Out Indication.		
	Ring 1 = 2, 3, 4, 9 , 11, & 12		
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13		
29.	GET phaseStatusGroupPhaseOns.2		
30.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2	Pass/Fail	
	AND $0x01 = 0x01$		
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	Note: Ensure that Max Out occurred on Phase 9.		
31.	Set HILL Detector Input 1 = Off and 7 = Off		
32.	DELAY .2 Seconds		
Teardown		[	
	PERFORM Detector Teardown – TC020 if not proceeding to another		
	detector operation test case		
	Test Case Results		
Tested By:	Date Tested	Pass/Fail	
Test Case Notes:	<notes></notes>	-	
Version History:	v1.00 04/12/06 Initial Draft – RDR		
	v1.01 07/05/06 Updated notes – RDR		
	v1.02 07/20/06 Implemented script and proofed – JJ		

## Detector 2 Operations

Test Case:	Title:	Detector 2 Operations	
TC002	Description:	Verifies the operation of Detector 2 to call and exte	nd Phase 2
	Constants:		
	Variables:	currentRedClear	
		currentYellowChange	
	Pass/Fail	The DUT shall pass every verification step included	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detect	or Operations Setup – TC019 if not already done	
	SO.		
Detector 2 calls Ph	ase 2 during 1+6 C	ireen	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus.	$1 \neq 0x03$ AND ringStatus.2 $\neq 0x03$	
3.	DELAY 1 Seco	nd	
4.	GET ringStatus	.1, ringStatus.2	
5.	WEND		
	Note: Loop until co	ntroller rests in green somewhere.	
6.	Set HIL Detector	Input 6 = On	
1.	DELAY .2 Seconds		
8.	GET phaseStatus	GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatu	IsGroupPhaseOns.1 ≠ 0x21 AND	
40		PhaseOns.2 ≠ 0x00	
10.	DELAY 1 Secol		
11.	GET phaseStat	usGroupPhaseOns.1,	
10		SupenaseOns.2	
12.	VVEND	atrollar rapping 1+6	
12		$100 \text{ Intermediates } 1 \neq 0.$	
11	DELAY 2 Second		
14.	Sot HITL Dotoctor		
10.	DELAY 2 Seconde	(2 full seconds)	
10.	GET phase Status	(2 Iuli Seconds)	
12		ONISE VALUE nhaseStatusGroupVehCalls 1	Dass/Fail
10.	$\Delta ND \Omega n \Omega = \Omega n \Omega$		rass/raii
	Note: Ensure that I	Phase 2 is NOT registering a Vehicle Call during 6	
	Green without a De	etector 2 Call.	
19.	Set HITL Detector	Input $2 = On$	
20.	DELAY 2 Seconds	(2 full seconds)	
21.	GET phaseStatus(	GroupVehCalls.1	
22.	VERIEY that RESE	PONSE VALUE phaseStatusGroupVehCalls.1	Pass/Fail
	AND 0x02 = 0x02		
	Note: Ensure that I	Phase 2 is registering a Vehicle Call during the	
	Green of Phase 6	with a Detector 2 Call.	
23.	Set HITL Detector	Input 6 and Input 2 = Off	
24.	DELAY 2 Seconds	(2 full seconds)	

25.	GET phaseStatusGroupVehCalls.1		
26	VERIEV that RESPONSE VALUE phaseStatusGroupVehCalls 1		
20.	$\Delta ND D D D D D D D D D D D D D D D D D D$	1 433/1 41	
	Note: Ensure that Phase 2 is NOT registering a Vahiele Call offer		
	Note. Ensure that Phase 2 is NOT registering a vehicle Call alter		
~=	Detector 2 Call Is removed.		
27.	GET phaseRedClear.6 = [currentRedClear] and		
	phaseYellowChange.6 = [currentYellowChange]		
28.	SET phaseRedClear.6 = 70 and phaseYellowChange.6 = 70		
	Note: Increase Red Clear and Yellow Change time to ensure enough		
	time for vehicle calls to register on controller during those phases.		
29.	Set HITL Detector Input 7 = On		
	'		
	Note: This causes 1+6 to advance to Yellow.		
30	DELAY 2 Seconds		
31	Set HITL Detector Input $7 = \Omega ff$		
20	DELAV 2 Seconde		
JZ.	DELAT 2 Seconds		
33.	GET ringStatus.2		
34.	WHILE ringStatus.2 AND 0x07 ≠ 0x04		
35.	DELAY 1 Second		
36.	GET ringStatus.2		
37.	WEND		
	Note: Wait for 6 Yellow.		
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1		
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13		
38	GET phaseStatusGroupVehCalls 1		
30	VERIEV that RESPONSE \/ALLE nhaseStatusGroup\/ebCalls 1	Pass/Fail	
00.	$\Delta ND 0x02 = 0x00$	1 433/1 41	
	Note: Ensure that Dhase 2 is NOT registering a Vahiala Call during		
	Phase 6 Valley without a Datastar 2 Call		
10	Priase o Fellow Williout a Delector 2 Call.		
40.	Set HILL Detector input 2 = On		
41.	DELAY 2 Seconds (Give it some time to make sure it is registered)		
42.	GET phaseStatusGroupVehCalls.1		
43.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1	Pass/Fail	
	AND $0x02 = 0x02$		
	Note: Ensure that Phase 2 is registering a Vehicle Call during Phase		
	6 Yellow with a Detector 2 Call.		
44.	Set HITL Detector Input 2 = Off		
45.	DELAY 2 Seconds (Give it some time to make sure it is cleared)		
46	GET phaseStatusGroupVehCalls 1		
47	VERIEV that RESPONSE VALUE nhaseStatusGrounVehCalls 1	Pass/Fail	
77.	$\Delta ND 0x02 = 0x00$		
	Noto: Ensure that Phase 2 is NOT registering a Vehicle Call offer		
	Note: Ensure that Fhase 2 is NOT registering a vehicle Call aller		
4.0			
48.	GET ringStatus.2		
49.			
-	WHILE ringStatus.2 AND 0x07 ≠ 0x05		
50.	WHILE ringStatus.2 AND 0x07 ≠ 0x05 DELAY 1 Second		

	WEND	
	<i>Note:</i> Wait for 6 Red (CBS = $xxxxx101$ ).	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
<b>F</b> 0	Ring $2 = 15, 16, 5, 6, 7, 8, & 13$	
53.		D/E-1
54.		Pass/Fail
	AND 0x02 - 0x00	
	Note: Ensure that Phase 2 is NOT registering a Vehicle Call during	
	Phase 6 Red without a Detector 2 Call.	
55.	Set HITL Detector Input 2 = On	
56.	DELAY 2 Seconds (Give it some time to make sure it is registered)	
57.	GET phaseStatusGroupVehCalls.1	
58.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1	Pass/Fail
	AND 0x02 = 0x02	
	Note: Ensure that Phase 2 is registering a Vehicle Call during the	
	Phase 6 Red with a Detector 2 Call.	
59.	Set HITL Detector Input 2 = Off	
60.	DELAY 2 Seconds (Give it some time to make sure it is cleared)	
61.	GET phaseStatusGroupVehCalls.1	
62.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1	Pass/Fail
	AND $0x02 = 0x00$	
	Note: Ensure that Dhese 2 is NOT registering a Vahiele Call offer the	
	Note: Ensure that Phase 2 is NOT registering a vehicle Call alter the	
63	SET phasePedClear 6 = [currentPedClear] and	
05.	screened by the second se	
Detector 2 calls Ph	pase 2 during 1+7 Green	
1.	GET ringStatus 1, ringStatus 2	
2	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
	$VI HEE HHUSIAIUS, I \neq UXUS AND HHUSIAIUS, Z \neq UXUS$	
3.	DELAY 1 Second	
3.	DELAY 1 Second GET ringStatus.1, ringStatus.2	
3. 4. 5.	DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
3. 4. 5.	DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
3. 4. 5.	DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere.	
3. 4. 5. 6.	DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On	
3. 4. 5. 6. 7.	DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds	
3. 4. 5. 6. 7. 8.	WHILE Hingstatus.1 ≠ 0x03 AND Hingstatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
3. 4. 5. 6. 7. 8. 9.	WHILE Hingstatus.1 ≠ 0x03 AND Hingstatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
3. 4. 5. 6. 7. 8. 9.	WHILE Hingstatus.1 ≠ 0x03 AND Hingstatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00	
3. 4. 5. 6. 7. 8. 9. 10.	WHILE Hingstatus.1 ≠ 0x03 AND Hingstatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second	
3. 4. 5. 6. 7. 8. 9. 10. 11.	WHILE Hingstatus.1 ≠ 0x03 AND Hingstatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,	
3. 4. 5. 6. 7. 8. 9. 10. 11.	WHILE Hingstatus.1 ≠ 0x03 AND Hingstatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2	
3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	WHILE Hingstatus.1 ≠ 0x03 AND Hingstatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2	
3.         4.         5.         6.         7.         8.         9.         10.         11.         12.	WHILE InigStatus.1 ≠ 0x03 AND InigStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches or is in 1+7.	
3.         4.         5.         6.         7.         8.         9.         10.         11.         12.         13.	WHILE InigStatus.1 ≠ 0x03 AND InigStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches or is in 1+7.         Set HITL Detector Input 7 = Off	
3.         4.         5.         6.         7.         8.         9.         10.         11.         12.         13.         14.	WHILE Hingstatus.1 ≠ 0x03 AND Hingstatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches or is in 1+7.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         QUELAY .2 Seconds	
3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15.	<pre>White hingstatus.1 ≠ 0x03 AND hingstatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches or is in 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On DELAY .2 Seconds </pre>	
3.         3.         4.         5.         6.         7.         8.         9.         10.         11.         12.         13.         14.         15.         16.	WHILE InigStatus.1 ≠ 0x03 AND HingStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches or is in 1+7.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         Set HITL Detector Input 2 = On         DELAY 2 Seconds (2 full seconds)	
3.         4.         5.         6.         7.         8.         9.         10.         11.         12.         13.         14.         15.         16.         17.	WHILE HigStatus.1 ≠ 0x03 AND HigStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches or is in 1+7.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         Set HITL Detector Input 2 = On         DELAY 2 Seconds (2 full seconds)         GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, meas	

18.	WHILE phaseStatusGroupGreens.1 = 0x41 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
10	phaseStatusGroupVehCalls.1 AND 0x02 ≠ 0x02	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupGreens.2,	
ົ ົ		
<u> </u>		
٢٢.	vERIF F phaseStatusGroupGreens. I = 0x4 FAND	Pass/Fall
	phaseStatusGroupVebCalls 1 AND 0x02 - 0x02	
	<i>Note</i> : Ensure that Phase 2 registers a Vehicle Call during 1+7 green.	
23.	Set HITL Detector Input 2 = Off	
24.	DELAY .2 Seconds	
Detector 2 calls Ph	ase 2 during 1+8 Green	
1.	GET ringStatus.1. ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1. ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVenCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x81 AND	
	phaseStatusGroupGreens.z = 0x00 AND phaseStatusGroupVobCalle 1 AND 0x02 + 0x02	
10	DELAV 1 Second	
<u> </u>	CET phaseStatusGroupGreens 1_phaseStatusGroupGreens 2	
20.	nhaseStatusGrounVehCalls 1	
21	WEND	
22	VERIEY phaseStatusGroupGreens 1 = 0x81 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	1 400/1 41
	phaseStatusGroupVehCalls.1 AND 0x02 = 0x02	
	· ····································	
	Note: Ensure that Phase 2 registers a Vehicle Call during 1+8 Green.	
23.	Set HITL Detector Input 2 = Off	
24.	DELAY .2 Seconds	
Detector 2 calls Ph	hase 2 during 1+13 Green	
1.	GET ringStatus.1, ringStatus.2	

2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6	Set HITL Detector Input 8 = On	
7	DELAY 2 Seconds	
8	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
0. Q	WHILE phaseStatusGroupPhaseOns $1 \neq 0.81$ AND	
Э.	$r_{\text{DaseStatusGroupPhaseOns}} 2 \neq 0x01 \text{AND}$	
10	DELAV 1 Second	
10.	CET phono Status Croup Phono One 1	
11.	GET pridseolalusGroupPhaseOns.1,	
10		
12.	VVEND Noto Wait until controller reaches 1 / 9	
40		
13.	Set HILL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HIL Detector Input 3 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x01 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x10	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 1+13.	
22.	Set HITL Detector Input 3 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 2 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x01 AND	
	phaseStatusGroupGreens.2 = 0x10 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 ≠ 0x02	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31	VERIEY phaseStatusGroupGreens 1 = 0x01 AND	Pass/Fail
•	phaseStatusGroupGreens 2 = 0x10 AND	
	phaseStatusGroupVehCalls 1 AND $0x02 = 0x02$	
	<i>Note:</i> Ensure that Phase 2 registers a Vehicle Call during 1+13	
	Green.	
32	Set HITL Detector Input 2 = Off	
33	DELAY 2 Seconds	
Detector 2 calls Ph	pase 2 during 3+5 Green	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
∠. 3	DEL ΔV 1 Second	
J.	CET ringStatus 1 ringStatus 2	
4.	GET THYSIAUS.T, THYSIAUS.Z	

5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x14 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 ≠ 0x02	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x14 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 = 0x02	
	Note: Ensure that Phase 2 registers a Venicle Call during 3+5 Green.	
23.	Set HILL Detector input $2 = Off$	
<u>24.</u>	DELAY .2 Seconds	
Detector 2 calls Pr	hase 2 during 4+5 Green	-
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 7 UXU3 AND ringStatus.2 7 UXU3	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Lean until controller reate in green comowhere	
6	Note. Loop until controller rests in green somewhere.	
0.	Set HITL Detector input 4 = On	
1.	DELAY .2 Seconds	
0.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	while phaseStatusGroupPhaseOns. 1 ≠ 0x18 AND	
10		
10.	CET phaseStatusGroupDhaseOne 1	
11.	GET phaseStatusGroupPhaseOns. 1,	
10		
12.	WEIND Noto: Wait until controllor roachae 4+5	
10	Note, wait until controller reaches 4+5.	
13.	DELAV 2 Seconda	
14.	DELAT .2 Seconds	
	So the contraction input $2 - 1$	

16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x18 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 ≠ 0x02	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 = 0x02	
	<i>Note:</i> Ensure that Phase 2 registers a Vehicle Call during 4+5 Green.	
23.	Set HITL Detector Input 2 = Off	
24.	DELAY .2 Seconds	
Detector 2 calls Ph	ase 2 during 6+12 Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
14.	DELAY 2 Seconds	
15.	Set HILL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
~	pnaseStatusGroupPhaseOns.2	
21.	WEND	
	INOTE: VValt Until Controller reaches 6+12.	
22.	Set HILL Detector Input 6 = Off	
23.	DELAY 2 Seconds	
24.	Set HILL Detector Input 2 = On	
25.	DELAY 2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	

27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2 = 0x08 AND phaseStatusGroupVehCalls 1 AND 0x02 ≠ 0x02	******
28	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls 1	
30	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2 = 0x08 AND phaseStatusGroupVehCalls.1 AND 0x02 = 0x02	Pass/Fail
	<i>Note:</i> Ensure that Phase 2 registers a Vehicle Call during 6+12 Green.	
32.	Set HITL Detector Input 2 = Off	
33.	DELAY .2 Seconds	
Detector 2 calls Ph	ase 2 during 6+11 Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x04	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
21	WEND	
۷۱.	Note: Wait until controller reaches 6+11	
22	Set HITL Detector Input 6 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 2 = On	
25.	DELAY .2 Seconds	
26	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens 2	
_0.	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 ≠ 0x02	

28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 = 0x02	
	Note: Ensure that Phase 2 registers a Vehicle Call during 6+11	
	Green.	
32.	Set HITL Detector Input 2 = Off	
33.	DELAY .2 Seconds	
Detector 2 calls Ph	ase 2 during 5+9 Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
6	Note: Loop until controller rests in green somewhere.	
0.	Set HILL Detector Input 3 = On	
/. o	DELAY .2 Seconds	
Ŏ.	GET phaseStatusGroupPhaseOns. T, phaseStatusGroupPhaseOns.2	
9.	while phasestatus Group Phase Ons. $1 \neq 0.014$ AND	
10	DELAV 1 Second	
10.	CET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
12		
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 7 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x10 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x01	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 5+9.	
22.	Set HITL Detector Input 7 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 2 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
07	pnaseStatusGroupVenCalls.1	
27.	while phaseStatusGroupGreens.1 = 0x10 AND	
	phaseStatusGroupVebCalls 1 AND $0x02 \neq 0x02$	
20	DELAV 1 Second	
<u>20.</u>	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
۷۶.	phaseStatusGroupVehCalls 1	

30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x10 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 = 0x02	
	Note: Ensure that Phase 2 registers a Vehicle Call during 5+9 Green.	
	Set HITL Detector Input 2 = Off	
33.	DELAY .2 Seconds	
Detector 2 extends	Phase 2 when Phase 2 is Green	-
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 2 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 6 = On	
14.	DELAY .2 Seconds	
15.	GET ringStatus.1	
16.	WHILE ringStatus.1 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
17.	DELAY 1 Second	
18.	GET ringStatus.1	
19.	WEND	
	Note: Wait for Max Out Indication.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
20.	GET phaseStatusGroupPhaseOns.1	
21.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x02 = 0x02	
	Note: Ensure that Max Out occurred on Phase 2.	
22.	Set HITL Detector Input 2 = Off and 6 = Off	
23.	DELAY .2 Seconds	

Teardown				
	PERFORM Detector Teardown -	TC020 if n	not proceeding to another	
	detector operation test case			
Test Case Results				
Tested By:		Date Tested		Pass/Fail
Test Case Notes:	<notes></notes>			
Version History:	v1.00 04/12/06 Initial Draft (Calls 6) – RDR v1.01 07/05/06 Updated notes – v1.02 07/20/06 Implemented scr	s during Ye RDR ript and pro	ellow and Red are only test pofed – JJ	ed for Phase

## Detector 3 Operations

Test Case:	Title:	Detector 3 Operations	
TC003	Description:	Verifies the operation of Detector 3 to call and exte	nd Phase 3
		under specific conditions and to extend interval 35	16B.
	Constants:		
	Variables:		
	Pass/Fail	The DUT shall pass every verification step included	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detector	or Operations Setup – TC019 if not already done	
	SO.		
Detector 3 calls Ph	ase 3 when Phase	3 and Phase 11 are not Green (In 1+6 Green)	
1.	GET ringStatus.1, r	ringStatus.2	
2.	WHILE ringStatus.	1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Secor	nd	
4.	GET ringStatus.	.1, ringStatus.2	
5.	WEND		
	Note: Loop until co	ntroller rests in green somewhere.	
6.	Set HITL Detector	Input 6 = On	
7.	DELAY .2 Seconds		
8.	GET phaseStatus	GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatu	sGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroup	PhaseOns.2 ≠ 0x00	
10.	DELAY 1 Secor	nd	
11.	GET phaseState	usGroupPhaseOns.1,	
	phaseStatusGro	pupPhaseOns.2	
12.	WEND		
	Note: Wait until cor	ntroller reaches 1+6.	
13.	Set HITL Detector	nput 6 = Off	
14.	DELAY .2 Seconds	· · · · · · · · · · · · · · · · · · ·	
15.	Set HITL Detector	Input 3 = On	
16.	DELAY .2 Seconds		
17.	GET phaseStatusG	GroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroup	VehCalls.1	

10		
18.	while phaseStatusGroupGreens.1 = 0x21 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls 1 AND 0x04 ≠ 0x04	
19.	DELAY 1 Second	
20	GET phaseStatusGroupGreens 1 phaseStatusGroupGreens 2	
20.	on phase of a function of the former of the	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x21 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls 1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 1+6 Green	
	Sot UITL Detector Input 2 - Off	
ZJ.		
24.	DELAY .2 Seconds	
Detector 3 calls Ph	hase 3 when Phase 3 and Phase 11 are not Green (In 1+7 Green)	
1.	GET ringStatus.1, ringStatus.2	
2	WHILE ringStatus $1 \neq 0x03$ AND ringStatus $2 \neq 0x03$	
3	DELAV 1 Second	
J	CET ringStatue 1 ringStatue 2	
4.		
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
0.	WHILE $phaseStatusCroupPhaseOps 1 \neq 0x41 AND$	
9.	nhaaoStatuaCrounDhaaoOna 2 d 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7	
12	Set HITL Detector Input $7 = \Omega ff$	
10.	DELAX 2 Casanda	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 3 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x41 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	nhaseStatusGroupVehCalls 1 AND $0x04 \neq 0x04$	
10	DELAV 1 Second	
	CET phoseStatusCroupCroops 1, phoseStatusCroupCroops 2	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	pnaseStatusGroupVenCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x41 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 1+7 Green	
23	Set HITI Detector Input 3 = Off	
20.	DELAV 2 Seconde	
۲4.		

Detector 3 calls Ph	ase 3 when Phase 3 and Phase 11 are not Green (In 1+8 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 3 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x81 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x81 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 1+8 green.	
23.	Set HIL Detector Input 3 = Off	
24.	DELAY 2 Seconds	
Detector 3 calls Pr	ase 3 when Phase 3 and Phase 11 are not Green (In 2+16 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller resis in green somewhere.	
0. 7	Set HIL Detector input $\delta = On$	
/.	DELAT .2 Seconds	
ŏ.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	whiLe phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
40	pnaseStatusGroupPnaseOns.2 ≠ 0X00	
10.	DELATISECOND	1

11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
18	WHILE phaseStatusGroupPhaseOns $1 \neq 0x02$ AND	
101	phaseStatusGroupPhaseOns.2 ≠ 0x80	
19	DFLAY 1 Second	
20	GET phaseStatusGroupPhaseOns 1	
20.	phaseStatusGroupPhaseOns 2	
21	WEND	
21.		
	Note: Wait until controller reaches 2+16	
22	Set HITL Detector Input $2 = Off$	
23	DELAY 2 Seconds	
20.	Set HITL Detector Input 3 = On	
27.	DELAY 2 Seconds	
20.	GET phaseStatusGroupGreens 1 phaseStatusGroupGreens 2	
20.	of the phase Status Group Veb Calls 1	
	WHILE phaseStatusGroupGroops 1 = 0x02 AND	
21.	nhasoStatusGroupGroops 2 = 0x80 AND	
	phaseStatusGroupVebCalle 1 AND $0x04 \neq 0x04$	
28	DELAV 1 Second	
20.	CET phaseStatueGroupGroops 1, phaseStatusGroupGroops 2	
29.	DET phaseStatusGroupSteens.1, phaseStatusGroupGteens.2,	
20		
31	VEDIEV phaseStatusGroupGreeps 1 - 0x02 AND	Dass/Fail
51.	nhaseStatusCroupGreens 2 - 0x80 AND	F 855/F 811
	phaseStatusGroupVebCalls 1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 2+16	
	areen	
32	Set HITL Detector Input 3 = Off	
33	DELAY 2 Seconds	
Detector 3 calls Ph	pase 3 when Phase 3 and Phase 11 are not Green (In 2+15 Green)	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
3	DELAY 1 Second	
<u> </u>	GET ringStatus 1_ringStatus 2	
5	WEND	
0.	WEND	
	Note: Loop until controller rests in green somewhere	
6	Set HITI. Detector Input $7 = \Omega n$	
7	DELAV 2 Seconde	
۲. و	GET nhaseStatusGrounPhaseOne 1_nhaseStatusGrounPhaseOne 2	
<u>٥</u> .	$\Box$	
9.	while phaseolalusoloup haseolis. $1 \neq 0.041$ AND haseolis $2 \neq 0.00$	
10	DELAY 1 Second	

11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY 2 Seconds	
15	Set HITL Detector Input $2 = On$	
16	DELAY 2 Seconds	
17	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
17.	WHILE phaseStatusGroupPhaseOne 1 $\pm$ 0v02 AND	
10.	nhaseStatusGrounPhaseOns $2 \pm 0x40$	
10	DELAV 1 Second	
20	GET phaseStatusCroupPhaseOne 1	
20.	obaseStatusGroupPhaseOns 2	
۷۱.	WEND	
	Noto: Mait until controllor reaches 2+15	
<u></u>	Note. Walt until controller reaches 2+15.	
<u> </u>	DELAV 2 Secondo	
<u>کې</u>	DELAT .2 Seconds	
24.	Set HITL Detector input 3 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
~	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 2+15	
32.	Set HIL Detector Input 3 = Off	
33.	DELAY .2 Seconds	
Detector 3 calls Pr	hase 3 when Phase 3 and Phase 11 are not Green (In 1+13 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	

11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15	Set HITL Detector Input $2 = On$	
16	DELAY 2 Seconds	
17	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
17.	WHILE phaseStatusGroupPhaseOne 1 $\pm$ 0v02 AND	
10.	nhaseStatusGrounPhaseOns $2 \pm 0x40$	
10	DELAV 1 Second	
20	GET phaseStatusCroupPhaseOne 1	
20.	obaseStatusGroupPhaseOns 2	
۷۱.	WEND	
	Noto: Wait until controller reaches 2+15	
	Sot HITL Detector Input 2 - Off	
<u> </u>	DELAV 2 Secondo	
23.	DELAY .2 Seconds	
24.	Set HILL Detector Input 3 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 2+15	
	green.	
32.	Set HIL Detector Input 3 = Off	
33.	DELAY .2 Seconds	
Detector 3 calls Pr	hase 3 when Phase 3 and Phase 11 are not Green (In 2+5 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 2 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	

11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 3 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x12 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1. phaseStatusGroupGreens.2.	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 2+5 Green.	
23.	Set HITL Detector Input 3 = Off	Pass/Fail
24	DELAY 2 Seconds	
Detector 3 calls Ph	ase 3 when Phase 3 and Phase 11 are not Green ( In 4+5 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus $1 \neq 0x03$ AND ringStatus $2 \neq 0x03$	
3	DELAY 1 Second	
4	GET ringStatus 1 ringStatus 2	
5	WEND	
0.		
	Note: Loop until controller rests in green somewhere.	
6	Set HITL Detector Input 4 = On	
7	DELAY 2 Seconds	
8	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
0. Q	WHILE phaseStatusGroupPhaseOne 1 $\pm$ 0v18 AND	
5.	nhaseStatusGroupPhaseOns $2 \pm 0x00$	
10		
10.	DELAY 1 Second	
11	DELAY 1 Second	
11.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
11.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
11. 12.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	
11.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	
11. 12.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 4+5.	
11. 12. 13.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 4+5. Set HITL Detector Input 4 = Off DELAY 2 Seconds	
11. 12. 13. 14.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND <i>Note:</i> Wait until controller reaches 4+5. Set HITL Detector Input 4 = Off DELAY .2 Seconds Set HITL Detector Input 3 = On	
11. 12. 13. 14. 15. 16	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND <i>Note:</i> Wait until controller reaches 4+5. Set HITL Detector Input 4 = Off DELAY .2 Seconds Set HITL Detector Input 3 = On DELAY .2 Seconds	
11. 12. 13. 14. 15. 16. 17	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 4+5. Set HITL Detector Input 4 = Off DELAY .2 Seconds Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupGreens 1, phaseStatusGroupGreens 2	
11. 12. 13. 14. 15. 16. 17.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 4+5. Set HITL Detector Input 4 = Off DELAY .2 Seconds Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVebCalls 1	
11. 11. 12. 13. 14. 15. 16. 17.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 4+5. Set HITL Detector Input 4 = Off DELAY .2 Seconds Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1 WHILE phaseStatusGroupGreens.1 = 0x18 AND	
11. 12. 13. 14. 15. 16. 17. 18.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND <i>Note:</i> Wait until controller reaches 4+5. Set HITL Detector Input 4 = Off DELAY .2 Seconds Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1 WHILE phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND	
11. 12. 13. 14. 15. 16. 17. 18.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 4+5. Set HITL Detector Input 4 = Off DELAY .2 Seconds Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1 WHILE phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls 1 AND 0x04 ± 0x04	
11. 11. 12. 13. 14. 15. 16. 17. 18. 19.	DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND <i>Note:</i> Wait until controller reaches 4+5. Set HITL Detector Input 4 = Off DELAY .2 Seconds Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1 WHILE phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04 DELAY .1 Second	

20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 4+5 Green.	
23.	Set HITL Detector Input 3 = Off	
24.	DELAY .2 Seconds	
Detector 3 calls Ph	hase 3 when Phase 3 and Phase 11 are not Green (In 6+12 Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
<u> </u>	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Meit until controller reaches 6 / 12	
<u></u>	Note. Walt until controller reaches 6+12.	
<u> </u>	Set HIL Delector Input 6 - On	
23.	Set HITL Detector Input 2 = On	
<u> </u>	DELAV 2 Seconds	
20. De	DELAT .2 3000005	
∠0.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	WHILE nhaseStatusCrounCreens 1 - 0v20 AND	
۷۱.	while phaseolalusoloupoleens. $I = 0X20$ AND phaseolatusCroupCroops $2 = 0x08$ AND	
	phaseotatusOloupOleetis.2 - 0000 AND haseoStatusOroupNahCalls 1 AND 0x00 + 0x00	
20	DELAV 1 Second	
20.	CET phaseStatusGroupGreens 1 phaseStatusGroupGroops 2	
2۶.	nhaseStatusGrounVehCalls 1	
L		

30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x08 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 6+12	
	Green.	
32.	Set HITL Detector Input 3 = Off	
33.	DELAY .2 Seconds	
Detector 3 calls Ph	ase 3 when Phase 3 and Phase 11 are not Green (In 5+9 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	No (c. Loop sustil controller note in succession companyle and	
~	Note: Loop until controller rests in green somewhere.	
<u> </u>	Set HILL Detector Input 4 = On	
<u> </u>	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
40	pnaseStatusGroupPnaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
10	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5	
12	Note: Walt until controller reaches $4+5$ .	
13.	DELAY 2 Seconds	
15	Set HITI Detector Input 7 = On	
10.	DELAY 2 Seconds	
10.	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
17.	WHILE nhaseStatusGroupPhaseOns 1 $\pm$ 0x10 AND	
10.	nhaseStatusGrounPhaseOns $2 \neq 0x01$	
19	DELAY 1 Second	
20	GET phaseStatusGroupPhaseOns 1	
_0.	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 5+9.	
22.	Set HITL Detector Input 7 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 3 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x10 AND	
	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31	VERIEY phaseStatusGroupGreens 1 = 0x10 AND	
---------------------------	------------------------------------------------------------------------------------------------------	---
•	nhaseStatusGrounGreens 2 = 0x01 AND	
	phaseStatusGroupVehCalls 1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 5+9 Green	
30	Set HITL Detector Input $3 = Off$	
22	DELAX 2 Seconde	
Detector 2 oxtende	DELAT .2 Seconds	
	CET ringStatus 1, ringStatus 2	
۱. ۲	GET IIIIgOtatus. I, IIIIgOtatus.2 M/LIII E ringStatus. 1 $\neq$ 0v02 AND ringStatus.2 $\neq$ 0v02	
<u>ک.</u>	DELAV 1 Second	
<u> </u>	DELAY I Second	
4. F		
5.	WEND	
	Material and until an atraller mastering and an announder a	
	Note: Loop until controller rests in green somewhere.	
<u> </u>	Set HILL Detector Input 3 = On	
<u> </u>	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 6 = On	
14.	DELAY .2 Seconds	
15.	GET ringStatus.1	
16.	WHILE ringStatus.1 AND 0x10 ≠= 0x10 (xxx1xxxx = maxout)	
17.	DELAY 1 Second	
18.	GET ringStatus.1	
19.	WEND	
	Note: Wait for Max Out Indication.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
20.	GET phaseStatusGroupPhaseOns.1	
21.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	
	AND 0x04 = 0x04	
	Note: Ensure that Max Out occurred on Phase 3	
22.	Set HITL Detector Input 3 = Off and 6 = Off	
23.	DELAY .2 Seconds	
<b>Detector 3 extends</b>	s Phase 11 during Phase 11	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
<b>.</b>		
	Note: Loop until controller rests in areen somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
· ·		•

8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
_	phaseStatusGroupPhaseOns.2 ≠ 0x04	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1.	
_	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 6+11.	
22.	Set HITL Detector Input 6 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 3 = On	
25.	DELAY .2 Seconds	
26.	Set HITL Detector Input 4 = On	
27.	DELAY .2 Seconds	
28.	GET ringStatus.1	
29.	WHILE ringStatus.1 AND $0x10 \neq 0x10$ (xxx1xxxx = maxout)	
30.	DELAY 1 Second	
31.	GET ringStatus.1	
32.	WEND	
	Note: Wait for Max Out Indication.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
33.	GET phaseStatusGroupPhaseOns.2	
34.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2	
	AND 0x04 = 0x04	
	Note: Ensure that Max Out occurred on Phase 11	
35.	Set HITL Detector Input 3 = Off and 4 = Off	
36.	DELAY .2 Seconds	
Teardown		
	PERFORM Detector Teardown – TC020 if not proceeding to another	
	detector operation test case	
	Test Case Results	
Tested By:	Date	Pass/Fail
	Tested	
Test Case Notes:	<notes></notes>	
Version History:	v1.00 04/14/06 Initial Draft – RDR	
-	v1.01 07/05/06 Updated notes – RDR	
	v1.02 07/19/06 Implemented script and proofed – JJ	

## Detector 4 Operations

Test Case:	Title:	Detector 4 Operations	
TC004	Description: Verifies the operation of Detector 4 to call and extend Phase 4		
	under specific conditions and to extend interval 4516B.		
	Constants:	·	
	Variables:		
	Pass/Fail	The DUT shall pass every verification step included	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup	•		
-	PERFORM Detect	or Operations Setup – TC019 if not already done	
	SO.		
Detector 4 calls Ph	ase 4 when Phase	4 and Phase 12 are not Green (In 1+6 Green)	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus.	1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Seco	nd	
4.	GET ringStatus	.1, ringStatus.2	
5.	WEND		
	Note: Loop until co	ntroller rests in green somewhere.	
6.	Set HITL Detector	Input 6 = On	
7.	DELAY .2 Seconds	3	
8.	GET phaseStatus	GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatu	IsGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroup	PhaseOns.2 ≠ 0x00	
10.	DELAY 1 Seco	nd	
11.	GET phaseStat	usGroupPhaseOns.1,	
	phaseStatusGr	pupPhaseOns.2	
12.	WEND		
	Note: Wait until co	ntroller reaches 1+6.	
13.	Set HITL Detector	Input 6 = Off	
14.	DELAY .2 Seconds	3	
15.	Set HITL Detector	Input 4 = On	
16.	DELAY .2 Seconds	6	
17.	GET phaseStatus	GroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroup	VehCalls.1	
18.	WHILE phaseStatu	IsGroupGreens.1 = 0x21 AND	
	phaseStatusGroup	Greens.2 = 0x00 AND	
	phaseStatusGroup	VehCalls.1 AND 0x08 ≠ 0x08	
19.	DELAY 1 Seco	nd	
20.	GET phaseStat	usGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGr	pupVehCalls.1	
21.	WEND		
22.	VERIFY phaseStat	tusGroupGreens.1 = 0x21 AND	Pass/Fail
	phaseStatusGroup	Greens.2 = 0x00  AND	
	pnaseStatusGroup	vencalis.1 AND 0x08 = 0x08	
	Note: Ensure that I	Phone 4 registers a Vahiela Call during 4+0 Course	
		-nase 4 registers a venicle Call during 1+6 Green.	
<u> </u>			
<b>∠</b> 4.	DELAY 2 Seconds	j	

Detector 4 calls Ph	hase 4 when Phase 4 and Phase 12 are not Green (In 1+7 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 4 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x41 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVenCalls.1	
21.		
22.	VERIFY phaseStatusGroupGreens.1 = 0x41 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVenCalls. TAND 0x06 - 0x06	
	Note: Ensure that Phase 4 registers a Vehicle Call during 1+7 Green	
23	Set HITI Detector Input $4 = Off$	
20.	DELAY 2 Seconds	
Detector 4 calls Ph	pase 4 when Phase 4 and Phase 12 are not Green (In 1+8 Green)	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03	
3	DELAY 1 Second	
4	GET ringStatus 1 ringStatus 2	
5	WEND	
0.		
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	

11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 4 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x81 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x81 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
	<i>Note</i> : Ensure that Phase 4 registers a Vehicle Call during 1+8 Green.	
23.	Set HITL Detector Input 4 = Off	
24.	DELAY .2 Seconds	
Detector 4 calls Ph	hase 4 when Phase 4 and Phase 12 are not Green (In 2+16 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	No (al and until combroller reads in success companyle and	
	Note: Loop until controller resis in green somewhere.	
0.	Set HILL Detector Input 8 = On	
1.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	pnaseStatusGroupPnaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Noto: Mait until controller reaches 1+9	
12	Note: Walt until controller reaches $1 \pm 0$ .	
17	DELAX 2 Secondo	
14.	DELAT .2 Seconds	
10. 16	DELAV 2 Seconde	
10.	DELAT .2 Seconds	
17.	G = 1 phase status of oup Thase Ons. 1, phase status of oup Thase Ons.2	
Ið.	while phaseolalusoloupmaseOns. $1 \neq 0x02$ AND phaseOstatusOroupPhaseOns. $2 \neq 0x00$	
10		
18. 20	DELAT I Second	
∠∪.	GET pridsestatusGroupPridseOffs.1,	
L	ทาลระองสนรงเงนุทศาสระบาร.2	

21.	WEND	
	Note: Wait until controller reaches 2+16.	
22.	Set HILL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HILL Detector Input 4 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x80 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x80 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
	<i>Note</i> : Ensure that Phase 4 registers a Vehicle Call during 2+16.	
~~~	Green.	
32.	Set HIL Detector Input 4 = Off	
33.	DELAY 2 Seconds	
Detector 4 calls Pr	hase 4 when Phase 4 and Phase 12 are not Green (in 2+15 Green)	
<u>ا.</u>	GET IIIIgStatus. 1, IIIIgStatus.2	
<u>∠.</u>	DELAV 1 Second	
ے۔ ۸	DELAT I Second	
<u>4.</u> Б		
5.	WEND	
	Note: Loop until controller rests in green somewhere	
6	Set HITL Detector Input 7 = On	
7	DELAY 2 Seconds	
8	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
9	WHILE phaseStatusGroupPhaseOns $1 \neq 0x41$ AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	

21.	WEND	
	Note: Wait until controller reaches 2+15.	
22.	Set HIL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HILL Detector Input 4 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVenCalls.1 AND 0x08 = 0x08	
	Note: Ensure that Dhase 4 registers a Vahiala Call during 2+15	
	Creen	
20	Set HITL Detector Input 4 - Off	
32.	DELAV 2 Seconds	
Detector 4 calls Ph	DELAT .2 Seconds	
	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 \pm 0x03 AND ringStatus 2 \pm 0x03	
2. 	DELAY 1 Second	
ـــــــــــــــــــــــــــــــــــــ	GET ringStatus 1 ringStatus 2	
5	WEND	
0.		
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	Set HILL Detector Input 3 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	vvHILE phaseStatusGroupPhaseOns.1 ≠ 0x01 AND	
	pnaseStatusGroupPnaseOns.2 ≠ 0x10	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	pnaseStatusGroupPnaseOns.2	

21.	WEND	
	Note: Wait until controller reaches 1+13.	
22.	Set HILL Detector Input 3 = Off	
23.	DELAY .2 Seconds	
24.	Set HILL Detector Input 4 = On	
25.	DELAY .2 Seconds	
20.	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x01 AND	
	phaseStatusGroupGreens.2 = 0x10 AND	
<u></u>	DELAX 1 Second	
<u> </u>	DELAT I Second	
29.	nhaseStatusGroupVehCalls 1	
30	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x01 AND	Pass/Fail
•	phaseStatusGroupGreens.2 = 0x10 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
	<i>Note</i> : Ensure that Phase 4 registers a Vehicle Call during 1+13	
	Green.	
32.	Set HILL Detector Input 4 = Off	
33. Detector A collo Di	DELAY .2 Seconds	
	ase 4 when Phase 4 and Phase 12 are not Green (In 2+5 Green)	
ا. م	GET IIIIgStatus, I, IIIIgStatus, Z	
<u>ک.</u>	DELAX 1 Second	
3. 1	CET ringStatus 1 ringStatus 2	
<u>4.</u> 5	WEND	
0.		
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 2 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Nata Mait until controller recebes 2 5	
10	Note: Wall until controller reaches 2+5.	
13.	Set HIL Delector Input 2 - On	
14.	DELAT .2 Seconds	
10. 16	DELAV 2 Seconde	
10.	CET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
17.	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x12 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
19.	DELAY 1 Second	

20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
	<i>Note</i> : Ensure that Phase 4 registers a Vehicle Call during 2+5 Green.	
23.	Set HITL Detector Input 4 = Off	
24.	DELAY .2 Seconds	
Detector 4 calls Ph	hase 4 when Phase 4 and Phase 12 are not Green (In 3+5 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	No tool and subtle controller made in success conservitions	
~	Note: Loop until controller rests in green somewhere.	
<u> </u>	Set HILL Detector Input 3 = On	
<u> </u>	DELAY .2 Seconds	
<u> </u>	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	whiLE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
10		
10.	CET phaseStatueCroupPhaseOne 1	
11.	GET phaseStatusGroupPhaseOns.1,	
10		
12.	WEND	
	Note: Wait until controller reaches 3+5	
13	Set HITL Detector Input $3 = Off$	
10.	DELAY 2 Seconds	
15	Set HITL Detector Input 4 = On	
16.	DELAY 2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x14 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x14 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
~~~	Note: Ensure that Phase 4 registers a Vehicle Call during 3+5 Green.	
23.	Set HILL Detector Input 4 = Off	
24.	DELAY .2 Seconds	
Detector 4 calls Ph	hase 4 when Phase 4 and Phase 12 are not Green (In 6+11 Green)	
<u>ا</u> .	GET HINGSTATUS. 1, FINGSTATUS.2	
<u>∠.</u>	VITILE IIIIGSTATUS.T 7 UXU3 AND TINGSTATUS.2 7 UXU3	
<u> </u>	DELAT 1 Second	
4.	GET ringStatus.1, ringStatus.2	

Note: Loop until controller rests in green somewhere.       6.     Set HITL Detector Input 3 = On       7.     DELAY .2 Seconds	
Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector Input 3 = On         7.       DELAY .2 Seconds	
6.     Set HITL Detector Input 3 = On       7.     DELAY .2 Seconds	
7. DELAY .2 Seconds	
8. GE I phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
<ol> <li>WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND</li> </ol>	
phaseStatusGroupPhaseOns.2 ≠ 0x00	
10. DELAY 1 Second	
11. GET phaseStatusGroupPhaseOns.1,	
phaseStatusGroupPhaseOns.2	
12. WEND	
Note: Wait until controller reaches 3+5.	
13. Set HITL Detector Input 3 = Off	
14. DELAY .2 Seconds	
15. Set HITL Detector Input 6 = On	
16. DELAY .2 Seconds	
17. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
phaseStatusGroupPhaseOns.2 ≠ 0x04	
19. DELAY 1 Second	
20. GET phaseStatusGroupPhaseOns.1,	
phaseStatusGroupPhaseOns.2	
21. WEND	
Note: Wait until controller reaches 6+11.	
22. Set HITL Detector Input 6 = Off	
23. DELAY .2 Seconds	
24. Set HITL Detector Input 4 = On	
25. DELAY .2 Seconds	
26. GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
phaseStatusGroupVehCalls.1	
27. WHILE phaseStatusGroupGreens.1 = 0x20 AND	
phaseStatusGroupGreens.2 = 0x04 AND	
phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
28. DELAY 1 Second	
29. GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
phaseStatusGroupVehCalls.1	
30. WEND	
31. VERIFY phaseStatusGroupGreens.1 = 0x20 AND Pass/Fail	
phaseStatusGroupGreens.2 = 0x04 AND	
phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
Note: Ensure that Phase 4 registers a Vehicle Call during 6+11	
Green.	
32. Set HITL Detector Input 4 = Off	
33. DELAY .2 Seconds	
Detector 4 calls Phase 4 when Phase 4 and Phase 12 are not Green (In 5+9 Green)	
1. GET ringStatus.1, ringStatus.2	
2. WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3. DELAY 1 Second	
4. GET ringStatus.1. ringStatus.2	

5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 7 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x10 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x01	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 5+9.	
22.	Set HITL Detector Input 7 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 4 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x10 AND	
	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x10 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
	Note: Ensure that Phase 4 registers a Vehicle Call during 5+9 Green.	
32.	Set HITL Detector Input 4 = Off	
33.	DELAY .2 Seconds	······
Detector 4 extends	s Phase 4 during Phase 4	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	

5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HILL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
10	pnaseStatusGroupPnaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
10	pnaseStatusGroupPnaseOns.2	
12.	WEND	
	Nata: Mait until controllar reaches 4+5	
10	Note. Walt until controller reaches 4+5.	
10.	DELAV 2 Seconda	
14.	DELAT .2 Seconds	
10.	GET IIIgoldus. I WEILE ringStatus 1 AND $0x10 \neq 0x10$ (xxx1xxxx = maxout)	
10.	$\frac{1}{1} = \frac{1}{1} = \frac{1}$	
17.	CET ringStatue 1	
10.		
19.	WEND	
	Note: Wait for Max Out Indication	
	Ring $1 = 2 3 4 9 11 12 \& 1$	
	Ring $2 = 15, 16, 5, 6, 7, 8, 8, 13$	
20	GET nhaseStatusGrounPhaseOns 1	
20.	VERIEV that RESPONSE VALUE phaseStatusGroupPhaseOns 1	Pass/Fail
	AND 0x08 = 0x08	
	Note: Ensure that Max Out occurred on Phase 4.	
22.	Set HITL Detector Input 4 = Off and 6 = Off	
23.	DELAY .2 Seconds	
Detector 4 extends	s Phase 12 during Phase 12	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	

~		
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 6+12.	
22.	Set HITL Detector Input 6 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 4 = On	
25.	DELAY .2 Seconds	
26.	Set HITL Detector Input 3 = On	
27.	DELAY .2 Seconds	
28.	GET ringStatus.1	
29.	WHILE ringStatus.1 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
30.	DELAY 1 Second	
31.	GET ringStatus.1	
32.	WEND	
	Note: Wait for Max Out Indication.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
33.	GET phaseStatusGroupPhaseOns.2	
34.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2	Pass/Fail
	AND 0x04 = 0x04	
	Note: Ensure that Max Out occurred on Phase 12.	
35.	Set HITL Detector Input 4 = Off and 3 = Off	
36.	DELAY .2 Seconds	
Teardown		
	PERFORM Detector Teardown – TC020 if not proceeding to another	
	detector operation test case	
	Test Case Results	
	Date	
Tested By:	Tested	
Test Case Notes:		
Version History:	v1.00 04/14/06 Initial Draft – RDR	
	v1.01 07/05/06 Updated notes – RDR	
	v1.02 07/20/06 Implemented script and proofed – JJ	

## Detector 5 Operations

Test Case:	Title:	Detector 5 Operations	
TC005	Description:	Verifies the operation of Detector 5 to call Phase 2	under
		specific conditions and extend intervals 1625B, 163	35, 1645B,
		1735B, 1745B, 1835B, and 1845B.	, ,
	Constants:		
	Variables:		
	Pass/Fail	The DUT shall pass every verification step included	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detect	or Operations Setup – TC019 if not already done	
	SO.		
<b>Detector 5 No Call</b>	on Phase 2 when 0	Overlap B is Green	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus.	1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Seco	nd	
4.	GET ringStatus	.1, ringStatus.2	
5.	WEND		
	Note: Loop until co	ntroller rests somewhere.	
6.	Set HITL Detector	Input 3 = On	
7.	DELAY .2 Seconds	8	
8.	GET phaseStatus	GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatu	usGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroup	PhaseOns.2 ≠ 0x00	
10.	DELAY 1 Seco	nd	
11.	GET phaseStat	usGroupPhaseOns.1,	
	phaseStatusGr	oupPhaseOns.2	
12.	WEND		
	Note: Wait until co	ntroller reaches 3+5.	
13.	Set HITL Detector	Input 3 = Off	
14.	DELAY .2 Seconds	5	
15.	GET ringStatus.1,	ringStatus.2	
16.	WHILE ringStatus.	1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
17.	DELAY 1 Seco	nd	
18.	GET ringStatus	.1, ringStatus.2	
19.	WEND		
	Note: Wait for 3+5	Green Rest.	
20.	GET overlapStatus	GroupGreens.1	
	Note: overlapStatu	sGroupGreens is optional and a GET may return	
	a noSuchName. T	his also assumes that Overlap $B = 5+6+X$ .	
21.	IF RESPONSE EF	ROR = noError THEN	
22.	VERIFY that R	ESPONSE VALUE overlapStatusGroupGreens.1	
	AND 0x02 = 0x	02	
	Note: Verifies that	Overlap B = Green.	
23.	END IF		

24.	Set HITL Detector Input 5 = On	
25.	DELAY 3 Seconds	
26.	GET phaseStatusGroupVehCalls.1. phaseStatusGroupVehCalls.2	
27.	VERIEV that RESPONSE VALUE phaseStatusGroupVehCalls 1	Pass/Fail
_/.	AND 0x02 = 0x00	
	Note: Verifies that no call in entered on Phase 2	
28	Set HITL Detector Input $5 = Off$	
29	DELAY 2 Seconds	
Detector 5 No Call	on Phase 2 when Overlan B is not Green and a Call on Phase 3	
1	GET ringStatus 1 ringStatus 2	
ו. ס	WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03	
2.	DELAV 1 Second	
J.	CET ringStatus 1. ringStatus 2	
4. <i>E</i>		
5.	WEND	
	Note: Loop until controller rests somewhere.	
b. 	Set HILL Detector Input / = On	
<u> </u>	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap $B = 5+6+X$ .	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND 0x02 = 0x00	
	<i>Note:</i> Verifies that Overlap B = NOT Green	
18.	END IF	
19	Set HITL Detector Input 3 = On	
20	DELAY 2 Seconds	
20.	Set HITL Detector Input 5 = On	
21.	DELAV 2 Seconds	
22.	GET nhaseStatusCroup\/ebCalls 1	
23.	VEDIEV that DESDONSE VALUE, phaseStatusCroupVabCalls 1	Doog/Eail
24.		Fass/Faii
	Note: Varifias that no call in antered on Dhase 2	
05	Note, veniles that no call in entered on Phase 2.	
25.	Set HILL Detector input $5 = O \pi$ and $3 = O \pi$	
26.	DELAY 2 Seconds	
Detector 5 No Call	on Phase 2 when Overlap B is not Green and a Call on Phase 4	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	

3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns 1	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8	
13	Set HITL Detector Input 8 = Off	
14	DELAY 2 Seconds	
15	GET overlanStatusGrounGreens 1	
10.		
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlan $B = 5+6+X$	
16	IF RESPONSE ERROR = $n_0$ Error THEN	
10.	VERIEV that RESPONSE VALUE overlapStatusGroupGreens 1	
17.	$\Delta ND 0x02 = 0x00$	
	AND 0X02 - 0X00	
	Note: Verifies that Overlap $B = NOT$ Green	
18	FND IF	
19	Set HITL Detector Input 4 = On	
20	DELAY 2 Seconds	
20.	Set HITL Detector Input 5 = On	
21.	DELAY 2 Seconds	
22.	GET nhaseStatusGroup\/ebCalls 1	
20.	VEDIEV that DESDONSE \/ALLIE phaseStatusCroup\/ebCalls 1	Dass/Fail
24.	$\Delta ND 0 = 0 \times 00$	r ass/r an
	AND 0x02 - 0x00	
	Note: Ensure that Phase 2 is not registering a Vehicle Call	
25	Set HITL Detector Input $4 = Off$ and $5 = Off$	
20.	DELAV 2 Seconds	
Detector 5 calls Ph	been 2 when Overlan B is not Green and no Calls on Phase 3 or Pha	so /
	GET ringStatus 1 ringStatus 2	
۱. ر	WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03	
2.	DELAV 1 Second	
J.	CET ringStatus 1. ringStatus 2	
4. E		
Э.	WEND	
	Noto: Loop until controllor rosts somowhoro	
6	Set HITL Detector Input 7 - On	
0.	DELAV 2 Secondo	
1.	DELAY .2 Seconds	
ŏ.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	vv⊓iLE pnaseStatusGroupPnaseOns.1 ≠ 0x41 AND	
	pnaseStatusGroupPnaseOns.2 ≠ 0x00	
10.	DELAY I SECOND	
11.	GET phaseStatusGroupPhaseOns.1,	

	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	
_		
	<i>Note:</i> overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap $B = 5+6+X$ .	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	
	AND 0x02 = 0x00	
	<i>Note:</i> Verifies that Overlap B = NOT Green	
18.	END IF	
19.	Set HITL Detector Input 5 = On	
20.	DELAY 2 Seconds	
21	GET phaseStatusGroupVehCalls 1	
22	VERIEV that RESPONSE VALUE phaseStatusGroupVehCalls 1	Pass/Fail
<i></i> .	AND 0x02 = 0x02	
	Note: Checks for call on Phase 2	
23	Set HITL Detector Input 5 = Off	
24	DELAY 2 Seconds	
Detector 5 extends	Phase 13 when Phase 13 is Green	
1.	GET ringStatus.1. ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3	DELAY 1 Second	
4	GET ringStatus 1 ringStatus 2	
5	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
9	WHILE phaseStatusGroupPhaseOns 1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 $\neq$ 0x00	
10.	DELAY 1 Second	
11	GET phaseStatusGroupPhaseOns 1	
	phaseStatusGroupPhaseOns 2	
12	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 4 = On	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 5 = On	
18	DELAY 2 Seconds	
19	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
20	WHILE phaseStatusGroupPhaseOns $1 \neq 0x01$ AND	
20.	phaseStatusGroupPhaseOns $2 \neq 0x10$	
21	DELAY 1 Second	
<b>— I</b> •		

*		
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WEND	
-0.		
	Note: Wait until controller reaches 1+13.	
24.	GET ringStatus.2	
25.	WHILE ringStatus.1 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
26.	DELAY 1 Second	
27.	GET ringStatus.2	
28.	WEND	
	Note: Wait for Max Out Indication.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
29.	GET phaseStatusGroupPhaseOns.1	
30.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2	Pass/Fail
	AND 0x10 = 0x10	
	Note: Ensure that Max Out occurred on Phase 13.	
31.	Set HITL Detector Input 5 = Off and 4 = Off	
32.	DELAY .2 Seconds	
Teardown		
	PERFORM Detector Teardown – TC020 if not proceeding to another	
	detector operation test case	
	Test Case Results	
	Date	Dece/Eail
Testeu by.	Tested	Pass/Fall
Test Case Notes:	<notes></notes>	
Version History:	v1.00 04/12/06 Initial Draft – RDR	
-	v1.01 07/05/06 Updated notes – RDR	
	v1.02 07/20/06 Implemented script and proofed – JJ	

## Detector 6 Operations

Test Case:	Title:	Detector 6 Operations	
TC006	Description:	Verifies the operation of Detector 6 to call and exte	nd Phase 6.
	Constants:		
	Variables:		
	Pass/Fail	The DUT shall pass every verification step include	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detect	or Operations Setup – TC019 if not already done	
	SO.		
Detector 6 calls Ph	nase 6 during 2+5 0	Green	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus.	.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Seco	nd	
4.	GET ringStatus	s.1, ringStatus.2	

5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 2 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x12 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 ≠ 0x20	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
۷۱.	WEND	
22	VEDIEV phaseStatusCroupCroops $1 = 0x12$ AND	Base/Eail
22.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND	Pass/Fail
22.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVebCalls 1 AND 0x20 = 0x20	Pass/Fail
22.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20	Pass/Fail
22.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green.	Pass/Fail
22.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off	Pass/Fail
22. 	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds	Pass/Fail
22. 23. 24. Detector 6 calls Pf	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds	Pass/Fail
22. 23. 24. Detector 6 calls Ph 1.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds ase 6 during 3+5 Green GET ringStatus.1, ringStatus.2	Pass/Fail
22. 23. 24. Detector 6 calls Ph 1. 2.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>nase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pr</b> 1. 2. 3.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	Pass/Fail
22. 23. 24. Detector 6 calls Pr 1. 2. 3. 4.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	Pass/Fail
22. 23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pr</b> 1. 2. 3. 4. 5.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>Tase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pf</b> 1. 2. 3. 4. 5.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere.	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pf</b> 1. 2. 3. 4. 5.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 3 = On	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pf</b> 1. 2. 3. 4. 5. 6. 7.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pr</b> 1. 2. 3. 4. 5. 5. 6. 7. 8.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pr</b> 1. 2. 3. 4. 5. 5. 6. 7. 8. 9.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>nase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pr</b> 1. 2. 3. 4. 5. 6. 7. 8. 9.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pf</b> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note</i> : Loop until controller rests somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pf</b> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, during StatusGroupPhaseOns.2	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pf</b> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note</i> : Loop until controller rests somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 $\neq$ 0x14 AND phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	Pass/Fail
22. 23. 24. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 Note: Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 $\neq$ 0x14 AND phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pf</b> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>nase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	Pass/Fail
22. 23. 24. <b>Detector 6 calls Pf</b> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 Note: Ensure that Phase 6 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>nase 6 during 3+5 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	Pass/Fail

14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x14 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 ≠ 0x20	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22	VERIEY phaseStatusGroupGreens 1 = 0x14 AND	Pass/Fail
	nhaseStatusGroupGreens 2 = 0x00 AND	r abb/r an
	phaseStatusGroupVehCalls 1 AND 0x20 = 0x20	
	Note: Ensure that Phase 6 registers a Vehicle Call during 3+5 Green	
23	Set HITL Detector Input 6 = Off	
20.	DELAY 2 Seconds	
Detector 6 calls Ph	base 6 during 4+5 Green	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
<u> </u>	DELAV 1 Second	
<u>J.</u>	CET ringStatus 1 ringStatus 2	
4. 5		
5.	WEND	
	Note: Loop until controller rests somewhere	
6	Set HITL Detector Input $4 = On$	
7	DELAY 2 Seconds	
8	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
0.	WHILE phaseStatusCroupPhaseOns 1 + 0v18 AND	
Э.	$r_{\text{haseStatusGroupPhaseOns}} 2 \neq 0.00$	
10	DELAV 1 Second	
11	GET phaseStatusGroupPhaseOns 1	
11.	ole i pilaseotatusOtouprilaseotis. i, phasoStatusOroupPhasoOne 2	
10		
12.	WEND	
	Note: Wait until controller reaches 4+5	
13	Set HITL Detector Input 4 - Off	
1.	DELAV 2 Seconde	
14.	Set HITL Detector Input 6 - On	
10.	DELAY 2 Secondo	
10.	DELAT .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
40		
18.	while phaseStatusGroupGreens.1 = 0x18 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	pnasestatusGroupvenCalls.1 AND 0x20 ≠ 0x20	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	pnaseStatusGroupVenCalls.1	
21.	WEND	

22.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 = 0x20	
	Note: Ensure that Phase 6 registers a Vehicle Call during 4+5 Green.	
23.	Set HITL Detector Input 6 = On	
24.	DELAY .2 Seconds	
Detector 6 calls Ph	nase 6 during 5+9 Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 7 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x10 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x01	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 5+9.	
22.	Set HITL Detector Input 7 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 6 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x10 AND	
	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 ≠ 0x20	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	

31.	VERIFY phaseStatusGroupGreens.1 = 0x10 AND phaseStatusGroupGreens.2 = 0x01 AND	Pass/Fail
	phaseStatusGroupVehCalls.1 AND 0x20 = 0x20	
	<i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 5+9 Green.	
32.	Set HITL Detector Input 6 = Off	
33.	DELAY .2 Seconds	
Detector 6 Calls Pl	nase 6 during 1+7 Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6	Set HITL Detector Input 7 = On	
7	DELAY 2 Seconds	
8	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
0. 0	WHILE phaseStatusGroupPhaseOne 1 $\pm$ 0v/1 AND	
9.	while phaseotalusoroup haseons $1 \neq 0.041$ AND phaseOtatusGroupPhaseOne $2 \neq 0.000$	
10	DELAV 1 Second	
10.	CET phaseStatusCroupPhaseOns 1	
11.	obasoStatusGroupPhasoOns 2	
10		
12.	WEND	
	Note: Wait until controller reaches or is in 1+7	
12	Sot HITL Detector Input 7 - Off	
11	DELAV 2 Seconde	
14.	Set HITL Detector Input 6 - On	
10.	Set HIL Detector Input 6 - On	
10.	DELAY .2 Seconds	
17.	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x41 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 ≠ 0x20	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x41 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 = 0x20	
	Note: Ensure that Phase 6 registers a Vehicle Call during 1+7 Green.	
23.	Set HITL Detector Input 6 = Off	
24.	DELAY .2 Seconds	
Detector 6 calls Ph	nase 6 during 1+8 Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	

6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11	GET phaseStatusGroupPhaseOns 1	
	phaseStatusGroupPhaseOns 2	
12		
	Note: Wait until controller reaches 1+8	
13	Set HITL Detector Input 8 = Off	
14	DELAY 2 Seconds	
15	Set HITL Detector Input $6 = On$	
16	DELAV 2 Seconds	
10.	GET phaseStatusGroupGreens 1 phaseStatusGroupGreens 2	
17.	baseStatusGroupVebCalls 1	
10	WHILE phaseStatusGroupGroops 1 = 0x81 AND	
10.	phaseStatusGroupGroops 2 = 0x00 AND	
	phaseStatusGroupVebCalle 1 AND $0x20 \neq 0x20$	
10	DELAV 1 Second	
<u> </u>	CET phono Status Croup Croops 1, phono Status Croup Croops 2	
20.	GET phaseStatusGroupSteens.1, phaseStatusGroupGreens.2,	
<u> </u>	VEND	Deco/Fail
٢٢.	vERIFT phaseStatusGroupGreens. 1 = 0x01 AND	Pass/Fall
	phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVabCalle 1 AND 0x20 = 0x20	
	phaseStatusGroupVenGalis. I AND 0x20 = 0x20	
	Note: Ensure that Dhase 6 registers a Vahiala Call during 1+9 Crean	
	<i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.	
23.	<i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 1+8 Green. Set HITL Detector Input 6 = Off	
23. 24.	<i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 1+8 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds	
23. 24. Detector 6 calls Ph	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds hase 6 during 2+16 Green	
23. 24. Detector 6 calls Pr 1.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 2+16 Green</b> GET ringStatus.1, ringStatus.2	
23. 24. Detector 6 calls Ph 1. 2.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds ase 6 during 2+16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
23. 24. Detector 6 calls Pr 1. 2. 3.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	
23. 24. Detector 6 calls Pr 1. 2. 3. 4.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         mase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2	
23. 24. <b>Detector 6 calls Pr</b> 1. 2. 3. 4. 5.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         ase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WHILE NUMBER         DELAY 1 Second         GET ringStatus.1, ringStatus.2	
23. 24. <b>Detector 6 calls Pr</b> 1. 2. 3. 4. 5.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         mase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1, ringStatus.2         WHILE NUMBER	
23. 24. Detector 6 calls Ph 1. 2. 3. 4. 5.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         ase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         mase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         mase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         mase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         ase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         ase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         ase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         ase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND         phaseStatusGroupPhaseOns.1, encode         GET phaseStatusGroupPhaseOns.1, encode	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         mase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2 ≠ 0x00	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 11.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         Dase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 11. 12.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds DELAY .2 Seconds DELAY ingStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.2, wence	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         DBLAY .2 Seconds         DELAY .1 Second         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+8.         Note: Wait until controller reaches 1+8.	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 11. 12. 13.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         DBLAY .2 Seconds         DELAY .1 Second         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+8.         Set HITL Detector Input 8 = Off	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12. 12. 13. 14.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         ase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND         phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+8.         Set HITL Detector Input 8 = Off         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, wenchase	
23. 24. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12. 12. 13. 14. 15.	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.         Set HITL Detector Input 6 = Off         DELAY .2 Seconds         ase 6 during 2+16 Green         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 8 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND         phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+8.         Set HITL Detector Input 8 = Off         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+8.         Set HITL Detector Input 8 = Off         DELAY .2 Seconds         Set HITL Detector Input 8 = Off         DELAY .2 Seconds         Set HITL Detector Input 2 = On	

17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x80	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 2+16.	
22.	Set HITL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 6 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens 1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x80 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 ≠ 0x20	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIEY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
•	phaseStatusGroupGreens.2 = 0x80 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 = 0x20	
	Note: Ensure that Phase 6 registers a Vehicle Call during 2+16	
	5 5	
	Green.	
32.	Green. Set HITL Detector Input 6 = Off	
32. 33.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds	
32. 33. Detector 6 calls Ph	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+15 Green	
32. 33. Detector 6 calls Pt 1.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds nase 6 during 2+15 Green GET ringStatus.1, ringStatus.2	
32. 33. Detector 6 calls Pt 1. 2.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
32. 33. Detector 6 calls Pt 1. 2. 3.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	
32. 33. Detector 6 calls Pt 1. 2. 3. 4.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
32. 33. Detector 6 calls Ph 1. 2. 3. 4. 5.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere.	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 7.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 7. 8.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds DELAY .2 Seconds DELAY .2 Seconds GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 6. 7. 8. 9.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds ase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 7. 8. 9.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds ase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 7. 8. 9. 9.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds DELAY .2 Seconds DELAY .2 Seconds GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second	
32. 33. Detector 6 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1,	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds ase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds mase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, WEND	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds DELAY .2 Seconds DELAY .2 Seconds GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 2+15 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, WEND <i>Note:</i> Wait until controller reaches 1+7.	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12. 12.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds DELAY .2 Seconds GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, WEND <i>Note:</i> Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off	
32. 33. Detector 6 calls Pt 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12. 12. 13. 14.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds DELAY .2 Seconds DELAY 15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, WEND Note: Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds	
32. 33. Detector 6 calls Pf 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 2+15 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, WEND <i>Note:</i> Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On	
32. 33. Detector 6 calls Pf 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12. 13. 14. 15. 16.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 2+15 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, WEND <i>Note:</i> Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On DELAY .2 Seconds	
32. 33. Detector 6 calls Pf 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17.	Green. Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>ase 6 during 2+15 Green</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, WEND <i>Note:</i> Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 2 = On DELAY .2 Seconds	

18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 2+15.	
22.	Set HITL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 6 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 ≠ 0x20	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 = 0x20	
	Note: Ensure that Phase 6 registers a Vehicle Call during 2+15	
	Green.	
32.	Set HITL Detector Input 6 = Off	
33.	DELAY .2 Seconds	
Detector 6 calls Pl	nase 6 during 1+13 Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	<i>ivote:</i> vvait until controller reaches 1+6.	
13.	Set HILL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	Set HILL Detector Input 2 = On	
16.	DELAY .2 Seconds	
	CET nhaseStatusGrounPhaseOns 1 nhaseStatusGrounPhaseOns 2	

18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x01 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x10	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 1+13.	
22.	Set HITL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 6 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
~_	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x01 AND	
	phaseStatusGroupGreens.2 = 0x10 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 ≠ 0x20	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x01 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x10 AND	
	phaseStatusGroupVenCalls.1 AND 0x20 = 0x20	
	Note: Ensure that Dhese 6 registers a Vahiala Call during 1,12	
	Note: Ensure that Phase 6 registers a vehicle Call during 1+13	
	Sof UIII Dotootor loout $G = Off$	
32.	Set HILL Detector Input 6 = Off	
32. 33.	Set HILL Detector Input 6 = Off DELAY .2 Seconds	
32. 33. Detector 6 extends	Set HILL Detector Input 6 = Off DELAY .2 Seconds S Phase 6 during 1+6 GET ringStatus 1 ringStatus 2	
32. 33. Detector 6 extends 1. 2	Set HILL Detector Input 6 = Off DELAY .2 Seconds <b>Phase 6 during 1+6</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus 1 ≠ 0x03 AND ringStatus 2 ≠ 0x03	
32. 33. Detector 6 extends 1. 2. 3	Set HILL Detector Input 6 = Off DELAY .2 Seconds S Phase 6 during 1+6 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	
32. 33. Detector 6 extends 1. 2. 3.	Set HITL Detector Input 6 = Off DELAY .2 Seconds Set FingStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus 1 ringStatus 2	
32. 33. Detector 6 extends 1. 2. 3. 4.	Set HITL Detector Input 6 = Off DELAY .2 Seconds S Phase 6 during 1+6 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
32. 33. Detector 6 extends 1. 2. 3. 4. 5.	Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>Phase 6 during 1+6</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
32. 33. Detector 6 extends 1. 2. 3. 4. 5.	Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>Phase 6 during 1+6</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere	
32. 33. Detector 6 extends 1. 2. 3. 4. 5.	Set HITL Detector Input 6 = Off DELAY .2 Seconds s Phase 6 during 1+6 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 6 = On	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6.	Set HITL Detector Input 6 = Off         DELAY .2 Seconds         S Phase 6 during 1+6         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests somewhere.         Set HITL Detector Input 6 = On         DELAY .2 Seconds	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8	Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>Phase 6 during 1+6</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9	Set HITL Detector Input 6 = Off DELAY .2 Seconds S Phase 6 during 1+6 GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9.	Set HITL Detector Input 6 = Off         DELAY .2 Seconds         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests somewhere.         Set HITL Detector Input 6 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns 2 ≠ 0x00	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.	Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>Phase 6 during 1+6</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>Phase 6 during 1+6</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1.	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>Phase 6 during 1+6</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	Set HITL Detector Input 6 = Off         DELAY .2 Seconds         SPhase 6 during 1+6         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests somewhere.         Set HITL Detector Input 6 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>Phase 6 during 1+6</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	Set HITL Detector Input 6 = Off         DELAY .2 Seconds         SPhase 6 during 1+6         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests somewhere.         Set HITL Detector Input 6 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2, ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2, ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2, wench = 0x21 AND         PhaseStatusGroupPhaseOns.2, ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2, wench = 0x21 AND         Note: Wait until controller reaches 1+6.	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13.	Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>Phase 6 during 1+6</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, WEND <i>Note:</i> Wait until controller reaches 1+6. Set HITL Detector Input 7 = On	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14.	Set HITL Detector Input 6 = Off DELAY .2 Seconds <b>Phase 6 during 1+6</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, WEND <i>Note:</i> Wait until controller reaches 1+6. Set HITL Detector Input 7 = On DELAY .2 Seconds	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15.	Set HITL Detector Input 6 = Off         DELAY .2 Seconds         Shase 6 during 1+6         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests somewhere.         Set HITL Detector Input 6 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET ringStatus.2	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16.	Set HITL Detector Input 6 = Off         DELAY .2 Seconds         SPhase 6 during 1+6         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests somewhere.         Set HITL Detector Input 6 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET ringStatus.2         WHILE ringStatus.2         WHILE ringStatus.2	
32. 33. Detector 6 extends 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17.	Set HITL Detector Input 6 = Off         DELAY .2 Seconds         SPhase 6 during 1+6         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests somewhere.         Set HITL Detector Input 6 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 7 = On         DELAY .2 Seconds         GET ringStatus.2         WHILE ringStatus.2 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)         DELAY 1 Second	

19.	WEND	
	Note: Wait for Max Out Indication.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
20.	GET phaseStatusGroupPhaseOns.1	
21.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 AND 0x20 = 0x20	Pass/Fail
	Note: Ensure that Max Out occurred on Phase 6.	
22.	Set HITL Detector Input 6 = Off and 7 = Off	
23.	DELAY .2 Seconds	
Teardown		
	PERFORM Detector Teardown – TC020 if not proceeding to another detector operation test case	
	Test Case Results	
Tested By:	Date Tested	
Test Case Notes:		
Version History:	v1.00 05/05/06 Initial Draft – RDR v1.01 07/05/06 Added 5+9, 2+16, 2+15, and 1+13. Updated notes – F v1.02 07/20/06 Implemented script and proofed – JJ	RDR

## Detector 7 Operations

Tast Casar	Title	Detector 7 Operations	
Test Case:	Detector / Operations		
1007	Description:	Verifies the operation of Detector 7 to call and extend Phase 7 under specific conditions and extend interval 1725B.	
	Constants:		
	Variables:		
	Pass/Fail	The DUT shall pass every verification step include	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detect	or Operations Setup – TC019 if not already done	
	SO.		
Detector 7 calls Ph	nase 7 when Phase	7 and Phase 15 are not Green (In 1+6 Green)	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus.	$1 \neq 0x03$ AND ringStatus. $2 \neq 0x03$	
3.	DELAY 1 Seco	nd	
4.	GET ringStatus	s.1, ringStatus.2	
5.	WEND		
	Note: Loop until co	ontroller rests in green somewhere.	
6.	Set HITL Detector	Input 6 = On	
7.	DELAY .2 Second	S	
8.	GET phaseStatus	GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseState	usGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroup	PhaseOns.2 ≠ 0x00	
10.	DELAY 1 Seco	nd	
11.	GET phaseStat	tusGroupPhaseOns.1,	
	phaseStatusGr	oupPhaseOns.2	

12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 7 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x21 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x21 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
	Note: Ensure that Phase 7 registers a Vehicle Call during 1+6 Green.	
23.	Set HITL Detector Input 7 = Off	
24.	DELAY .2 Seconds	
Detector 7 calls Ph	ase 7 when Phase 7 and Phase 15 are not Green (In 1+8 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
<u> </u>	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note the Mail with controller reaches 410	
10	Note: Walt until controller reaches 1+0.	
ای. ۱۵.	Set HITL Detector input $\delta = On$	
14.	DELAY .2 Seconds	
ID.	Set HILL Detector input / = On	
10.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
10		
18.		
	phaseStatusGroupGreens.2 = 0x00 AND	
	pnaseStatusGroupVenCalls.1 AND 0x40 ≠ 0x40	
19.	DELAY I Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupvenCalls.	

21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x81 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
	<i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 1+8 Green.	
23.	Set HITL Detector Input 7 = Off	
24.	DELAY .2 Seconds	
Detector 7 calls Ph	ase 7 when Phase 7 and Phase 15 are not Green (In 2+16 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HILL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	pnaseStatusGroupPnaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
10	pnaseStatusGroupPnaseOns.2	
12.	WEND	
	Noto: Wait until controller reaches 1+8	
12	Set HITL Detector Input 8 - Off	
11	DELAV 2 Seconds	
15	Set HITL Detector Input 2 = On	
16.	DELAY 2 Seconds	
10.	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
17.	WHILE nhaseStatusGroupPhaseOns $1 \neq 0x02$ AND	
10.	phaseStatusGroupPhaseOns $2 \neq 0x80$	
19	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 2+16.	
22.	Set HITL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 7 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x80 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail

	phaseStatusGroupGreens.2 = 0x80 AND phaseStatusGroupVebCalls 1 AND 0x40 = 0x40	
	<i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 2+16 Green.	
32.	Set HITL Detector Input 7 = Off	
33.	DELAY .2 Seconds	
Detector 7 calls Pl	nase 7 when Phase 7 and Phase 15 are not Green (In 1+13 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus 1 ≠ 0x03 AND ringStatus 2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 3 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x01 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x10	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Noto: Wait until controller reaches 1+13	
22	Set HITL Detector Input $3 = Off$	
22.	DELAX 2 Seconds	
23.	Set HITL Detector Input $7 = On$	
27.	DELAY 2 Seconds	
20.	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
20.	nhaseStatusGrounVehCalls 1	
27	WHILE phaseStatusGroupGreens 1 = 0x01 AND	
<b>_</b> /.	phaseStatusGroupGreens 2 = 0x10 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1. phaseStatusGroupGreens.2.	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x01 AND	Pass/Fail
	phaseStatusGroupGreens 2 = 0x10 AND	

	phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
	p	
	<i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 1+13	
	Green	
32.	Set HITL Detector Input 7 = Off	
33	DELAY 2 Seconds	
Detector 7 calls Ph	ase 7 when Phase 7 and Phase 15 are not Green (in 2+5 Green)	
1	GET ringStatus 1 ringStatus 2	
ו. ס	WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03	
2.	DELAV 1 Second	
3. 1	CET ringStatus 1 ringStatus 2	
4. E		
5.	WEND	
	Noto: Loop until controller roots in groop comowhere	
6	Note, Loop until controller resis in green somewhere.	
0.	Set HITL Detector input 2 = On	
<u> </u>	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 7 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x12 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
	h	
	Note: Ensure that Phase 7 registers a Vehicle Call during 2+5 Green.	
23.	Set HITL Detector Input 7 = Off	
24.	DELAY .2 Seconds	
Detector 7 calls Ph	ase 7 when Phase 7 and Phase 15 are not Green (In 3+5 Green)	
1.	GET ringStatus 1, ringStatus 2	
2	WHILE ringStatus $1 \neq 0x03$ AND ringStatus $2 \neq 0x03$	
3	DELAY 1 Second	
4	GET ringStatus 1 ringStatus 2	
5	WEND	
J.		
	Note: Loop until controller rests in green somewhere	
6	Set HITL Detector Input 3 = On	
ψ.		

7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 7 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x14 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x14 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.	
23.	<i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 3+5 Green. Set HITL Detector Input 7 = Off	
23. 24.	<i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 3+5 Green. Set HITL Detector Input 7 = Off DELAY .2 Seconds	
23. 24. Detector 7 calls Ph	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green. Set HITL Detector Input 7 = Off DELAY .2 Seconds ase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)	
23. 24. <b>Detector 7 calls Pr</b> 1.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green. Set HITL Detector Input 7 = Off DELAY .2 Seconds ase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2	
23. 24. <b>Detector 7 calls Ph</b> 1. 2.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.Set HITL Detector Input 7 = OffDELAY .2 Secondsase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)GET ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
23. 24. <b>Detector 7 calls Ph</b> 1. 2. 3.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         ase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second	
23. 24. <b>Detector 7 calls Ph</b> 1. 2. 3. 4.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green. Set HITL Detector Input 7 = Off DELAY .2 Seconds Tase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	
23. 24. Detector 7 calls Ph 1. 2. 3. 4. 5.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green. Set HITL Detector Input 7 = Off DELAY .2 Seconds mase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
23. 24. <b>Detector 7 calls Pr</b> 1. 2. 3. 4. 5.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green. Set HITL Detector Input 7 = Off DELAY .2 Seconds DELAY .2 Seconds DELAY .2 Seconds DET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         mase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.	
23. 24. Detector 7 calls Ph 1. 2. 3. 4. 5.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         ase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         mase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         ase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         ase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
23. 24. Detector 7 calls Ph 1. 2. 3. 4. 5. 6. 7. 8. 9.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         mase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         mase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         mase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND         phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         ase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND         phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, ≠ 0x00	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 11.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         ase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2, wEND	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 11. 12.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green. Set HITL Detector Input 7 = Off DELAY .2 Seconds TringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green. Set HITL Detector Input 7 = Off DELAY .2 Seconds DELAY .2 Seconds DELAY .2 Seconds GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 4+5. Set HITL Detector Input 4 = Off	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12. 13. 14.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         mase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 4+5.         Set HITL Detector Input 4 = Off         Net: Wait until controller reaches 4+5.         Set HITL Detector Input 4 = Off	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12. 13. 14.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         aase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WHLE ringStatus.1, ringStatus.2         WHILE ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 4+5.         Set HITL Detector Input 4 = Off         DELAY .2 Seconds         Set HITL Detector Input 4 = Off         DELAY .2 Seconds         Set HITL Detector Input 4 = Off         DELAY .2 Seconds         Set HITL Detector Input 4 = Off	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12. 13. 14. 15. 14.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         aase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, wence         WEND         Note: Wait until controller reaches 4+5.         Set HITL Detector Input 4 = Off         DELAY .2 Seconds         Set HITL Detector Input 4 = Off         DELAY .2 Seconds         Set HITL Detector Input 7 = On         DELAY .2 Seconds         Set HITL Detector Input 7 = On	
23. 24. Detector 7 calls Pr 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17.	Note: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green.         Set HITL Detector Input 7 = Off         DELAY .2 Seconds         mase 7 when Phase 7 and Phase 15 are not Green (In 4+5 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 4 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2, ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 4+5.         Set HITL Detector Input 4 = Off         DELAY .2 Seconds         Set HITL Detector Input 7 = On         DELAY .2 Seconds         Set HITL Detector Input 7 = On         DELAY .2 Seconds         Set HITL Detector Input 7 = On	

	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x18 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
19.	DELAY 1 Second	
20	GET phaseStatusGroupGreens 1 phaseStatusGroupGreens 2	
20.	phaseStatusGroupVehCalls 1	
21	WEND	
22	VERIEV phaseStatusGroupGreens 1 = 0x18 AND	Pass/Fail
<i>LL</i> .	nhaseStatusGrounGreens 2 = 0x00 AND	1 455/1 41
	phaseStatusGroupVehCalls 1 AND 0x40 = 0x40	
	Note: Ensure that Phase 7 registers a Vehicle Call during 4+5 Green	
23	Set HITL Detector Input 7 = Off	
20.	DELAY 2 Seconds	
Detector 7 calls Ph	ase 7 when Phase 7 and Phase 15 are not Green (In 6±12 Green)	
	GET ringStatus 1 ringStatus 2	
۱. ر	WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03	
2.	DELAV 1 Second	
J.	DELAT I SECOLU CET ringStatus 1. ringStatus 2.	
4. 5		
Э.	WEND	
	Note I can until controller rests in groon computers	
	Note: Loop unui controller resis in green somewhere.	
0.	Set HILL Detector Input 4 = On	
1.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 6+12.	
22.	Set HITL Detector Input 6 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 7 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens 1 = 0x20 AND	

phaseStatusGroupVehCalls: 1 AND 0x40 ≠ 0x40         28.       DELAY 1 Second         29.       GET phaseStatusGroupGreens: 1, phaseStatusGroupGreens: 2, phaseStatusGroupCreens: 2 = 0x08 AND phaseStatusGroupVehCalls: 1 AND 0x40 = 0x40         30.       WEN0         31.       VERIFY phaseStatusGroupGreens: 2 = 0x08 AND phaseStatusGroupVehCalls: 1 AND 0x40 = 0x40         Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green.         32.       Set HITL Detector Input 7 = Off         33.       DELAY 1 Seconds         Detector 7 calls Phase 7 when Phase 7 and Phase 15 are not Green (In 6+11 Green)         1.       GET ringStatus.1, ringStatus.2         2.       WHILE ingStatus.1 + 2 6x03 AND ringStatus.2 ≠ 0x03         3.       DELAY 1 Second         4.       GET ringStatus.1, ringStatus.2         5.       WEND         Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector input 3 = 0 n         7.       DELAY 1 Second         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         9.       WHILE phasStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         12.       WEND         Note: Loop until controller reaches 3+5.         13.       Set H		phaseStatusGroupGreens.2 = 0x08 AND	
28.       DELAY 1 Second         29.       GET phaseStatusGroupOreens.1, phaseStatusGroupOreens.2, phaseStatusGroupVehCalls.1         30.       WEND         31.       VERIFY phaseStatusGroupOreens.2 = 0x08 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40         Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green.       Pass/Fail         32.       Set HITL Detector Input 7 = Off         33.       DELAY 1 Second         Detector 7 calls Phase 7 when Phase 7 and Phase 15 are not Green (In 6+11 Green)         1.       GET ringStatus.1, ringStatus.2         2.       WHILE ringStatus.1, ringStatus.2         3.       DELAY 1 Second         4.       GET ringStatus.1, ringStatus.2         5.       WEND         Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector Input 3 = 0n         7.       DELAY 1 Second         8.       GET phaseStatusGroupPhaseOns.1 # 0x14 AND         phaseStatusGroupPhaseOns.2 # 0x00       DiaseStatusGroupPhaseOns.2 # 0x01         11.       GET phaseStatusGroupPhaseOns.2         12.       WEND       Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = 0ff         14.       DELAY 1 Second       DELAY 1 Second		phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
29.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         30.       WEND         31.       VERIFY phaseStatusGroupGreens.2 = 0x08 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40         Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green.       9         32.       Set HITL Detector Input 7 = Off         33.       DELAY 2 Seconds         Detector 7 calls Phase 7 when Phase 7 and Phase 15 are not Green (In 6+11 Green)       1         1.       GET ringStatus.1, ringStatus.2 ≠ 0x03         33.       DELAY 1 Second       1         4.       GET ringStatus.1, ningStatus.2 ≠ 0x03       1         5.       WEND       1       6         7.       DELAY 1 Second       1       1         6.       Set HITL Detector Input 3 = On       1       1         7.       DELAY 1 Second       1       1         8.       GET phaseStatusGroupPhaseOns.1 + 0x14 AND       1         9.       WHILE phaseStatusGroupPhaseOns.1 + 0x14 AND       1         9.       WHILE phaseStatusGroupPhaseOns.2       1         9.       WEND       1       1         10.       DELAY 1 Second       1       1         11.       GET phaseStatusGroupPhaseOns.2 <td< td=""><td>28.</td><td>DELAY 1 Second</td><td></td></td<>	28.	DELAY 1 Second	
30.       WEND         31.       VERFY phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40       Pass/Fail         31.       Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green.       Pass/Fail         32.       Set HITL Detector Input 7 = Off       Detector 7 calls Phase 7 when Phase 7 and Phase 15 are not Green (In 6+11 Green)         1.       GET ringStatus.1, ringStatus.2       Pass/Fail         2.       WHILE ringStatus.1 x # 0x03 AND ringStatus.2 # 0x03       Detector 7 calls Phase 7 when Phase 7 and Phase 15 are not Green (In 6+11 Green)         1.       GET ringStatus.1 x # 0x03 AND ringStatus.2 # 0x03       Detector 7 calls Phase 7 when Phase 7 and Phase 15 are not Green (In 6+11 Green)         3.       DELAY 1 Second       A.       GET ringStatus.1, ringStatus.2       S.         5.       WEND       Note: Loop until controller rests in green somewhere.       Set HITL Detector Input 3 = On       P.         7.       DELAY 1 Second       DELAY 1 Second       Detector 7 calls PhaseStatusGroupPhaseOns.1 # 0x14 AND       PhaseStatusGroupPhaseOns.2         9.       WHILE phaseStatusGroupPhaseOns.2       Note: Wait until controller reaches 3+5.       Set HITL Detector Input 3 = Off         11.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       Note: Wait until controller reaches 3+5.       Set HITL Detector Input 5 = Off         13.	29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	
31.     VERIFY phaseStatusGroupGreens. 1 = 0x20 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40     Pass/Fail       Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green.	30.	WEND	
phaseStatusGroupCreens.2 = 0x08 AND         phaseStatusGroupVehCalls.1 AND 0x40 = 0x40         Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         32.       Set HITL Detector Input 7 = Off         33.       DELAY 2.2 Seconds         Detector 7 calls Phase 7 when Phase 7 and Phase 15 are not Green (In 6+11 Green)         1.       GET ringStatus.1, ringStatus.2 ≠ 0x03         3.       DELAY 1 Second         4.       GET ringStatus.1, ringStatus.2         5.       WEND         Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector Input 3 = On         7.       DELAY 1 Second         8.       GET phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.2 ≠ 0x00         12.       WHIL E phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ± 0x00       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY.2 Seconds         15.       Set HITL Detector	31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fail
phaseStatusGroupVehCalls.1 AND 0x40 = 0x40Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green.32.Set HITL Detector Input 7 = Off33.DELAY.2 SecondsDetector 7 calls Phase 7 and Phase 15 are not Green (In 6+11 Green)1.GET ringStatus.1 ingStatus.22.WHILE ringStatus.1 4 x003 AND ringStatus.2 $\neq$ 0x033.DELAY 1 Second4.GET ringStatus.1, ringStatus.25.WENDNote: Loop until controller rests in green somewhere.6.Set HITL Detector Input 3 = On7.DELAY 1 Second8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE inaseStatusGroupPhaseOns.1 + 0x14 AND phaseStatusGroupPhaseOns.210.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.112.WENDNote: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY 2 Seconds15.Set HITL Detector Input 6 = On16.DELAY 2 Seconds17.GET phaseStatusGroupPhaseOns.1 + 0x20 AND phaseStatusGroupPhaseOns.2 $\neq$ 0x0419.DELAY 1 Second21.WEND22.Set HITL Detector Input 6 = Off23.DELAY 1 Second24.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.221.WEND22.Set HITL Detector Input 7 = On23.DELAY 1 Second24.Set HITL Detector Input 7 = On25.DELAY 2 Seconds </td <td></td> <td>phaseStatusGroupGreens.2 = 0x08 AND</td> <td></td>		phaseStatusGroupGreens.2 = 0x08 AND	
Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green.32.Set HITL Detector Input 7 = Off33.DELAY 2. SecondsDetector 7 calls Phase 7 when Phase 7 and Phase 15 are not Green (In 6+11 Green)1.GET ringStatus. 1, ringStatus.2 $\neq$ 0x033.DELAY 1. Second4.GET ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x035.WENDNote: Loop until controller rests in green somewhere.6.Set HITL Detector Input 3 = On7.DELAY 2. Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 $\neq$ 0x14 ANDphaseStatusGroupPhaseOns.2 $\neq$ 0x0010.DELAY 1. Second11.GET phaseStatusGroupPhaseOns.1.11.GET phaseStatusGroupPhaseOns.1.12.WENDNote: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY 2. Seconds17.GET phaseStatusGroupPhaseOns.1 $\neq$ 0x20 AND18.WHILE phaseStatusGroupPhaseOns.1 $\neq$ 0x20 ANDphaseStatusGroupPhaseOns.2 $\neq$ 0x0419.DELAY 2. Seconds11.GET phaseStatusGroupPhaseOns.1 $\neq$ 0x20 AND13.Set HITL Detector Input 6 = On14.DELAY 2. Seconds17.GET phaseStatusGroupPhaseOns.218.WHILE phaseStatusGroupPhaseOns.1 $\neq$ 0x20 AND19.DELAY 1. Second20.GET phaseStatusGroupPhaseOns.221.WENDNote		phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         32.       Set HITL Detector Input 7 = Off         33.       DELAY.2 Seconds         Detector 7 calls Phase 7 when Phase 7 and Phase 15 are not Green (In 6+11 Green)         1.       GET ringStatus.1, ringStatus.2         2.       WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         4.       GET ringStatus.1, ringStatus.2         5.       WEND         Note: Loop util controller rests in green somewhere.         6.       Set HITL Detector Input 3 = On         7.       DELAY 2 Seconds         8.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         9.       WHILE phaseStatusGroupPhaseOns.1 + 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00       DELAY 1 Second         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY 1 Second         15.       Set HITL Detector Input 3 = On         14.       DELAY 2 Seconds         15.       Set HITL Detector Input 3 = Off <td></td> <td></td> <td></td>			
Green.         32.       Set HITL Detector Input 7 = Off         33.       DELAY. 2 Seconds         Detector 7 calls Phase 7 when Phase 7 and Phase 15 are not Green (in 6+11 Green)         1.       GET ringStatus. 1, ringStatus.2         2.       WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         3.       DELAY 1 Second         4.       GET ringStatus.1, ringStatus.2         5.       WEND         Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector Input 3 = On         7.       DELAY.2 Seconds         8.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         9.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00       DELAY 1 Second         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2       DELAY 2 Seconds         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 6 = On         16.       DELAY 2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.2 + 0x04 AND         pha		Note: Ensure that Phase 7 registers a Vehicle Call during 6+12	
32.       Set HITL Detector Input 7 = Off         33.       DELAY 2 Seconds         Detector 7 calls Phase 7 when Phase 7 and Phase 15 are not Green (In 6+11 Green)         1.       GET ringStatus.1, ringStatus.2         2.       WHILE ringStatus.1 + 0x03 AND ringStatus.2 ≠ 0x03         3.       DELAY 1 Second         4.       GET ringStatus.1, ringStatus.2         5.       WEND         Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector Input 3 = 0n         7.       DELAY 2 Seconds         8.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         9.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY 2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY 2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 + 0x20 AND         phaseStatusGroupPhaseOns.		Green.	
33.       DELAY .2 Seconds         Detector 7 calls Phase 7 when Phase 15 are not Green (In 6+11 Green)         1.       GET ringStatus 1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         2.       WHILE ringStatus 1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         3.       DELAY 1 Second         4.       GET ringStatus 1, ringStatus 2         5.       WEND         Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector Input 3 = On         7.       DELAY 1 Second         8.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         9.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.1,         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY 1 Second         15.       Set HITL Detector Input 6 = On         16.       DELAY 2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND         phaseStatusGroupPhaseOns.2       DELAY 1 Second         20.       GET phaseStatusGroupPha	32.	Set HITL Detector Input 7 = Off	
Detector 7 calls Phase 7 and Phase 15 are not Green (In 6+11 Green)           1.         GET ringStatus 1, ringStatus 2           2.         WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03           3.         DELAY 1 Second           4.         GET ringStatus.1, ringStatus.2           5.         WEND           Note: Loop until controller rests in green somewhere.           6.         Set HITL Detector Input 3 = On           7.         DELAY 2. Seconds           8.         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2           9.         WHILE phaseStatusGroupPhaseOns.1 $\neq$ 0x14 AND phaseStatusGroupPhaseOns.2           9.         WHILE becond           11.         GET phaseStatusGroupPhaseOns.2           12.         WEND           Note: Wait until controller reaches 3+5.           13.         Set HITL Detector Input 3 = Off           14.         DELAY 2. Seconds           15.         Set HITL Detector Input 6 = On           16.         DELAY 2. Seconds           17.         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2           18.         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2           19.         DELAY 1. Second           20.         GET phaseStatusGroupPha	33.	DELAY .2 Seconds	
1.       GET ringStatus.1, ringStatus.2         2.       WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         3.       DELAY 1 Second         4.       GET ringStatus.1, ringStatus.2         5.       WEND         Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector Input 3 = 0n         7.       DELAY 2 Seconds         8.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         9.       WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY 2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY 2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1 ≠ 0x20 AND         phaseStatusGroupPhaseOns.2 ≠ 0x04       DelAY 1 Second         21.       WEND         Note: Wait until con	Detector 7 calls Ph	hase 7 when Phase 7 and Phase 15 are not Green (In 6+11 Green)	
2.       WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         3.       DELAY 1 Second         4.       GET ringStatus.1, ringStatus.2         5.       WEND         Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector Input 3 = On         7.       DELAY 2 Seconds         8.       GET phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY 2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY 2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off <t< td=""><td>1.</td><td>GET ringStatus.1, ringStatus.2</td><td></td></t<>	1.	GET ringStatus.1, ringStatus.2	
3.       DELAY 1 Second         4.       GET ringStatus.1, ringStatus.2         5.       WEND         Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector Input 3 = On         7.       DELAY 2 Seconds         8.       GET phaseStatusGroupPhaseOns.1 $\neq$ 0x14 AND phaseStatusGroupPhaseOns.2 $\neq$ 0x00         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.2 $\neq$ 0x00         11.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY 2 Seconds         15.       Set HITL Detector Input 3 = Off         14.       DELAY 2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 $\neq$ 0x20 AND         phaseStatusGroupPhaseOns.2 $\neq$ 0x04       DeLAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         19.       DELAY 1 Second       DeLAY 1 Second         20.       GET phaseStatusGroupPhaseOns.2       20.         21.       WEND       Note: Wait until controller reaches 6+11.	2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
<ul> <li>4. GET ringStatus.1, ringStatus.2</li> <li>5. WEND</li> <li>Note: Loop until controller rests in green somewhere.</li> <li>6. Set HITL Detector Input 3 = 0n</li> <li>7. DELAY .2 Seconds</li> <li>8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2</li> <li>9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00</li> <li>10. DELAY 1 Second</li> <li>11. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2</li> <li>12. WEND</li> <li>Note: Wait until controller reaches 3+5.</li> <li>13. Set HITL Detector Input 6 = On</li> <li>16. DELAY .2 Seconds</li> <li>17. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2</li> <li>18. WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2</li> <li>19. Note: Wait until controller reaches 3+5.</li> <li>13. Set HITL Detector Input 6 = On</li> <li>16. DELAY .2 Seconds</li> <li>17. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2</li> <li>18. WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2</li> <li>19. DELAY 1 Second</li> <li>20. GET phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2</li> <li>21. WEND</li> <li>Note: Wait until controller reaches 6+11.</li> <li>22. Set HITL Detector Input 6 = Off</li> <li>23. DELAY .2 Seconds</li> <li>24. Set HITL Detector Input 7 = On</li> <li>25. DELAY .2 Seconds</li> <li>26. GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupPhaseDate.2</li> <li>27. WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2, phaseStatusGroupPhaseDate.3</li> <li>27. WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2, phaseStatusGroupPhaseDate.3</li> </ul>	3.	DELAY 1 Second	
5.       WEND         Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector Input 3 = On         7.       DELAY .2 Seconds         8.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         9.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 × 0x00         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY .2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY 1.2 Seconds         24.       Set HITL Detector Input 7 = On         25.	4.	GET ringStatus.1, ringStatus.2	
Note: Loop until controller rests in green somewhere.         6.       Set HITL Detector Input 3 = On         7.       DELAY .2 Seconds         8.       GET phaseStatusGroupPhaseOns.1 # 0x14 AND         phaseStatusGroupPhaseOns.2 # 0x00       phaseStatusGroupPhaseOns.1 # 0x14 AND         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.2 # 0x00         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY .2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1 # 0x20 AND         phaseStatusGroupPhaseOns.2 # 0x04       PhaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       GET phaseStatusGroupPhaseOns.2         23.       DELAY 1 Second         24.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds	5.	WEND	
Note:       Loop Until Controller rests in green somewhere.         6.       Set HITL Detector Input 3 = On         7.       DELAY. 2 Seconds         8.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         9.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY.2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY.2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND         phaseStatusGroupPhaseOns.2 ≠ 0x04       DeLAY 1 Second         20.       GET phaseStatusGroupPhaseOns.2 ≠ 0x04         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY 1 Second         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGrou			
6.       Set HITL Detector Input 3 = 0n         7.       DELAY 2 Seconds         8.       GET phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00       phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY .2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY 1 Second         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupPhaseStatusGroupPhaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         <	~	Note: Loop until controller rests in green somewhere.	
7.       DELAY 2 Seconds         8.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         9.       WHILE phaseStatusGroupPhaseOns.1 $\neq$ 0x14 AND         phaseStatusGroupPhaseOns.2 $\neq$ 0x00       10.         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY 2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY 2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 $\neq$ 0x20 AND         phaseStatusGroupPhaseOns.2 $\neq$ 0x04       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY 1 Seconds       DELAY 2 Seconds         24.       Set HITL Detector Input 7 = On       DELAY 2 Seconds         25.       DELAY 2 Seconds       DELAY 2 Seconds	6.	Set HILL Detector Input 3 = On	
<ul> <li>8. Get phaseStatusGroupPhaseOns.1 + phaseStatusGroupPhaseOns.2</li> <li>9. WHILE phaseStatusGroupPhaseOns.1 + 0x14 NND phaseStatusGroupPhaseOns.2 + 0x00</li> <li>10. DELAY 1 Second</li> <li>11. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2</li> <li>12. WEND</li> <li>Note: Wait until controller reaches 3+5.</li> <li>13. Set HITL Detector Input 3 = Off</li> <li>14. DELAY .2 Seconds</li> <li>15. Set HITL Detector Input 6 = On</li> <li>16. DELAY .2 Seconds</li> <li>17. GET phaseStatusGroupPhaseOns.1 + 0x20 AND phaseStatusGroupPhaseOns.1 + 0x20 AND</li> <li>phaseStatusGroupPhaseOns.2 + 0x04</li> <li>19. DELAY 1 Second</li> <li>20. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2</li> <li>21. WEND</li> <li>Note: Wait until controller reaches 6+11.</li> <li>22. Set HITL Detector Input 6 = Off</li> <li>23. DELAY .2 Seconds</li> <li>24. Set HITL Detector Input 7 = On</li> <li>25. DELAY .2 Seconds</li> <li>26. GET phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupPhaseIns.1 = 0x20 AND</li> <li>27. WHILE phaseStatusGroupGreens.1 = 0x20 AND</li> <li>phaseStatusGroupPhaseIns.1 = 0x20 AND</li> </ul>	/.	DELAY .2 Seconds	
9.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY .2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND         phaseStatusGroupPhaseOns.2       0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND <td><u> </u></td> <td>GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2</td> <td></td>	<u> </u>	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
10.       DELAY 1 Second         11.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY .2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND         phaseStatusGroupPhaseOns.2 ≠ 0x04       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	9.	whiLE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
10.       DELAT 1 Second         11.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY .2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	10		
11.       OBET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         12.       WEND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY .2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	10.	CET phaseStatueCroupPhaseOne 1	
12.       WEND         12.       WeND         Note: Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY .2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2, phaseStatusGroupGreens.2 = 0x04 AND	11.	GET phaseStatusGroupPhaseOns 1,	
Note:       With U         Note:       Wait until controller reaches 3+5.         13.       Set HITL Detector Input 3 = Off         14.       DELAY .2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2, phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2, phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2 = 0x04 AND	12		
Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 6 = On16.DELAY .2 Seconds17.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.218.WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x0419.DELAY 1 Second20.GET phaseStatusGroupPhaseOns.221.WENDNote: Wait until controller reaches 6+11.22.Set HITL Detector Input 6 = Off23.DELAY .2 Seconds24.Set HITL Detector Input 7 = On25.DELAY .2 Seconds26.GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2, phaseStatusGroupGreens.2	12.	WEND	
13.       Set HITL Detector Input 3 = Off         14.       DELAY .2 Seconds         15.       Set HITL Detector Input 6 = On         16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2		Note: Wait until controller reaches 3+5.	
14.DELAY .2 Seconds15.Set HITL Detector Input 6 = On16.DELAY .2 Seconds17.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.218.WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x0419.DELAY 1 Second20.GET phaseStatusGroupPhaseOns.221.WENDNote: Wait until controller reaches 6+11.22.Set HITL Detector Input 6 = Off23.DELAY .2 Seconds24.Set HITL Detector Input 7 = On25.DELAY .2 Seconds26.GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.127.WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2 = 0x04 AND	13.	Set HITL Detector Input 3 = Off	
15.       Set HITL Detector Input 6 = On         16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupPresens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupVehCalls.1	14.	DELAY .2 Seconds	
16.       DELAY .2 Seconds         17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.2         21.       WEND         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2 = 0x04 AND	15.	Set HITL Detector Input 6 = On	
17.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND phaseStatusGroupPhaseOns.2 ≠ 0x04         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2 = 0x04 AND	16.	DELAY .2 Seconds	
18.       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND         19.       DELAY 1 Second         20.       GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2       phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2 = 0x04 AND	17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
phaseStatusGroupPhaseOns.2 ≠ 0x0419.DELAY 1 Second20.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.221.WENDNote: Wait until controller reaches 6+11.22.Set HITL Detector Input 6 = Off23.DELAY .2 Seconds24.Set HITL Detector Input 7 = On25.DELAY .2 Seconds26.GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.127.WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2 = 0x04 AND	18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
19.DELAY 1 Second20.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.221.WENDNote: Wait until controller reaches 6+11.22.Set HITL Detector Input 6 = Off23.DELAY .2 Seconds24.Set HITL Detector Input 7 = On25.DELAY .2 Seconds26.GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.127.WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens 2 = 0x04 AND		phaseStatusGroupPhaseOns.2 ≠ 0x04	
20.       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens 2 = 0x04 AND	19.	DELAY 1 Second	
phaseStatusGroupPhaseOns.2         21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2 = 0x04 AND	20.	GET phaseStatusGroupPhaseOns.1,	
21.       WEND         Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2 = 0x04 AND		phaseStatusGroupPhaseOns.2	
Note: Wait until controller reaches 6+11.22.Set HITL Detector Input 6 = Off23.DELAY .2 Seconds24.Set HITL Detector Input 7 = On25.DELAY .2 Seconds26.GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.127.WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens 2 = 0x04 AND	21.	WEND	
Note: Wait until controller reaches 6+11.         22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens 2 = 0x04 AND			
22.       Set HITL Detector Input 6 = Off         23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens 2 = 0x04 AND		Note: Wait until controller reaches 6+11.	
23.       DELAY .2 Seconds         24.       Set HITL Detector Input 7 = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens 2 = 0x04 AND	22.	Set HILL Detector Input 6 = Off	
24.       Set HILL Detector Input / = On         25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens 2 = 0x04 AND	23.	DELAY .2 Seconds	
25.       DELAY .2 Seconds         26.       GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1         27.       WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens 2 = 0x04 AND	24.	Set HILL Detector Input / = On	
20. GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1     27. WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens.2 = 0x04 AND	25.	DELAY .2 Seconds	
27. WHILE phaseStatusGroupGreens.1 = 0x20 AND phaseStatusGroupGreens 2 = 0x04 AND	26.	GET pnaseStatusGroupGreens.1, pnaseStatusGroupGreens.2,	
27. while phaseStatusGroupGreens 2 = 0x04 AND phaseStatusGroupGreens 2 = 0x04 AND	07		
	۷۱.	phaseStatusGroupGreens 2 = 0x04 AND	

	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
	Note: Ensure that Phase 7 registers a Vehicle Call during 6+11	
	Green.	
32.	Set HITL Detector Input 7 = Off	
33.	DELAY .2 Seconds	
Detector 7 calls Ph	nase 7 when Phase 7 and Phase 15 are not Green (In 5+9 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
	DELAY .2 Seconds	
15.	Set HITL Detector Input 8 = On	
16.	DELAY .2 Seconds	
	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x10 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x01	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Nata: Wait until controllar reaches 5+0	
	Note: Walt until controller reaches $5+9$ .	
<u> </u>	DELAV 2 Seconde	
23.	Sot HITL Detector Input 7 - On	
24.	DELAV 2 Seconds	
20. 06	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGroops 2	
20.	nhaseStatusGrounVehCalls 1	
27	WHILE nhaseStatusGrounGreens $1 = 0v10 \Delta ND$	
21.	nhaseStatusGrounGreens 2 = 0x01 AND	
	phaseStatusGroupVehCalls 1 AND $0x40 \neq 0x40$	
J		

28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x10 AND	Pass/Fail
-	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
	Note: Ensure that Phase 7 registers a Vehicle Call during 5+9 Green.	
32.	Set HITL Detector Input 7 = Off	
33.	DELAY .2 Seconds	
Detector 7 extends	Phase 7 during Phase 7	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 8 = On	
14.	DELAY .2 Seconds	
15.	GET ringStatus.2	
16.	WHILE ringStatus.2 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
17.	DELAY .2 Second	
18.	GET ringStatus.2	
19.	WEND	
	Note: Wait for Max Out Indication.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
20.	GET phaseStatusGroupPhaseOns.1	
21.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND $0x40 = 0x40$	
	Note Francisco that Marc October suggestion Disease 7	
	Note: Ensure that Max Out occurred on Phase 7.	
22.	Set HILL Detector input $7 = OIT$ and $8 = OIT$	
23.	DELAY .2 Seconds	
	CET ringStatus 1 ringStatus 2	
1.		
<u>∠.</u>	VITILE IIIIGSTATUS. T 7 UXU3 AND TINGSTATUS. 2 7 UXU3	
<u> </u>	DELAY 1 Second	
4.		
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
-----	------------------------------------------------------------	-----------
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 2+15.	
22.	Set HITL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 7 = On	
25.	DELAY .2 Seconds	
26.	Set HITL Detector Input 3 = On	
27.	DELAY .2 Seconds	
28.	GET ringStatus.2	
29.	WHILE ringStatus.2 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
30.	DELAY 1 Second	
31.	GET ringStatus.2	
32.	WEND	
-		
	Note: Wait for Max Out Indication.	
	Rina 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
33.	GET phaseStatusGroupPhaseOns.2	
34.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2	Pass/Fail
	AND 0x40 = 0x40	
	Note: Ensure that Max Out occurred on Phase 15.	
35.	Set HITL Detector Input 7 = Off and 3 = Off	
36.	DELAY .2 Seconds	

Teardown		
	PERFORM Detector Teardown – TC020 if not proceeding to another	
	detector operation test case	
Test Case Results		
Tested By:	Date Tested	
Test Case Notes:		
Version History:	v1.00 05/05/06 Initial Draft – RDR v1.01 07/05/06 Updated notes – RDR v1.02 07/20/06 Implemented script and proofed – JJ	

## Detector 8 Operations

Test Case:	Title:	Detector 8 Operations	
TC008	Description: Verifies the operation of Detector 8 to call and extend Phase 8		
		under specific conditions and extend interval 1825	3.
	Constants:	·	
	Variables:		
	Pass/Fail 1	The DUT shall pass every verification step included	d within the
	Criteria: T	Fest Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detector	Operations Setup – TC019 if not already done	
	SO.		
Detector 8 calls Ph	ase 8 when Phase 8	and Phase 16 are not Green (In 1+6 Green)	
1.	GET ringStatus.1, rin	ngStatus.2	
2.	WHILE ringStatus.1	≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second		
4.	GET ringStatus.1	, ringStatus.2	
5.	WEND		
	Note: Loop until cont	roller rests in green somewhere.	
6.	Set HITL Detector In	put 6 = On	
7.	DELAY .2 Seconds		
8.	GET phaseStatusGro	oupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatus	GroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupPl	haseOns.2 ≠ 0x00	
10.	DELAY 1 Second		
11.	GET phaseStatus	sGroupPhaseOns.1,	
	phaseStatusGrou	ipPhaseOns.2	
12.	WEND		
	Note: Wait until contr	roller reaches 1+6.	
13.	Set HITL Detector In	put 6 = Off	
14.	DELAY .2 Seconds		
15.	Set HITL Detector In	put 8 = On	
16.	DELAY .2 Seconds		
17.	GET phaseStatusGro	oupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVe	ehCalls.1	
18.	WHILE phaseStatus	GroupGreens.1 = 0x21 AND	
	phaseStatusGroupG	reens.2 = 0x00 AND	
	phaseStatusGroupVe	ehCalls.1 AND 0x80 ≠ 0x80	

19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x21 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 = 0x80	
	Note: Ensure that Phase 8 registers a Vehicle Call during 1+6 Green.	
23.	Set HIL Detector Input 8 = Off	
24.	DELAY .2 Seconds	
Detector 8 calls Pr	hase 8 when Phase 8 and Phase 16 are not Green (In 1+7 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Nota: Loop until controller rocts in groop comowhere	
6	Sot HITL Detector Input 7 = On	
7	DELAV 2 Seconds	
Q 7.	GET nhaseStatusGroupDhaseOns 1, nhaseStatusGroupDhaseOns 2	
0.	SET phase Status Group Phase Ons. 1, phase Status Group Phase Ons. 2	
9.	while phasestatus Group Phase Ons $2 \neq 0x00$	
10	DELAY 1 Second	
10.	GET nhaseStatusGrounPhaseOns 1	
	nhaseStatusGrounPhaseOns 2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = On	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 8 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x41 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 ≠ 0x80	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.		
22.	VERIFY phaseStatusGroupGreens.1 = 0x41 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVobCalle 1 AND 0x80 = 0x80	
	phasestatusoloupvenoalis. 1 AND 0x00 = 0x00	
	Note: Ensure that Phase 8 registers a Vehicle Call during 1+7 Green	
23	Set HITI Detector Input 8 = Off	
24.	DELAY .2 Seconds	
Detector 8 calls Pr	hase 8 when Phase 8 and Phase 16 are not Green (In 2+15 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	

4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
-		
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 7 = On	
7	DELAY 2 Seconds	
8	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
0.	WHILE phase Status Group Phase One $1 \neq 0x/11$ AND	
9.	while phase status of oup hase one $2 \neq 0.00$	
10	DELAV 1 Second	
10.	DELAT I Secoliu	
11.	GET phaseStatusGroupPhaseOns.1,	
	pnaseStatusGroupPnaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1.	
_ • •	phaseStatusGroupPhaseOns.2	
21	WEND	
	Note: Wait until controller reaches 2+15	
22	Set HITL Detector Input $2 = Off$	
22.	DELAV 2 Seconds	
20.	Sot HITL Detector Input 8 - On	
24.	DELAX 2 Secondo	
20.	CET phageStatusCroupCroops 1, phageStatusCroupCroops 2	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
~7		
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVenCalls.1 AND 0x80 ≠ 0x80	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 = 0x80	
	Note: Ensure that Phase 8 registers a Vehicle Call during 2+15	
	Green.	
32.	Set HITL Detector Input 8 = Off	
33.	DELAY .2 Seconds	
Detector 8 calls Ph	hase 8 when Phase 8 and Phase 16 are not Green (In 1+13 Green)	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03	
2. 2	DELAY 1 Second	
J. 1	CET ringStatue 1 ringStatue 2	
4.		

5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
18	WHILE phaseStatusGroupPhaseOns $1 \neq 0x01$ AND	
10.	phaseStatusGroupPhaseOns 2 ≠ 0x10	
19	DFLAY 1 Second	
20	GET phaseStatusGroupPhaseOns 1	
20.	phaseStatusGroupPhaseOns 2	
21	WEND	
21.		
	Note: Wait until controller reaches 1+13	
22	Set HITL Detector Input $2 = Off$	
23	DELAY 2 Seconds	
20.	Set HITL Detector Input 8 = On	
25	DELAY 2 Seconds	
20.	GET phaseStatusGroupGreens 1 phaseStatusGroupGreens 2	
20.	nhaseStatusGrounVehCalls 1	
27	WHILE phaseStatusGroupGreens 1 = 0x01 AND	
21.	nhaseStatusGrounGreens 2 = 0x10 AND	
	phaseStatusGroupVehCalls 1 AND $0x80 \neq 0x80$	
28	DELAY 1 Second	
20.	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
20.	nhaseStatusGrounVehCalls 1	
30	WEND	
31	VEDIEV phaseStatusGroupGreeps 1 - 0v01 AND	Dass/Eail
51.	phaseStatusCroupGreens 2 - 0x10 AND	F 855/F 811
	phaseStatusGroupVebCalls 1 AND 0x80 - 0x80	
	Note: Ensure that Phase 8 registers a Vehicle Call during 1+13	
	Green	
32	Set HITL Detector Input 8 = Off	
33	DELAY 2 Seconds	
Detector 8 calls Ph	use 8 when Phase 8 and Phase 16 are not Green (In 2+5 Green)	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
<u>د.</u> ع	DELAY 1 Second	
<u>.</u> Л	GET ringStatus 1 ringStatus 2	
4. E		
່ວ.		

~		
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 2 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	No to Mait until controllor processor Or E	
	Note: Walt until controller reaches 2+5.	
13.	Set HILL Detector Input 2 = Off	
14.	DELAY 2 Seconds	
15.	Set HILL Detector Input 8 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
40	pnaseStatusGroupVenCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x12 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	pnaseStatusGroupVenCalls.1 AND 0x80 ≠ 0x80	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	pnaseStatusGroupVenCalls.1	
21.		D / C - 11
22.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND	Pass/Fall
	phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVabCalle 1 AND 0x80 = 0x80	
	phaseStatusGroupVenCalls. TAND 0x00 = 0x00	
	Note: Ensure that Phase 8 registers a Vehicle Call during 2+5 Green	
22	Set HITL Detector Input 8 - Off	
23.	DELAV 2 Seconds	
Detector 8 calls Pr	ase 8 when Phase 8 and Phase 16 are not Green (in 3+5 Green)	
	GET ringStatus 1 ringStatus 2	
î	WHILE ringStatus 1 $\pm$ 0v03 AND ringStatus 2 $\pm$ 0v03	
2. 	DELAV 1 Second	
J.	GET ringStatus 1 ringStatus 2	
<del>_</del> 5	WEND	
0.	WEND	
	Note: Loop until controller rests in green somewhere	
6	Set HITL Detector Input 3 = On	
7	DELAY 2 Seconds	
8.	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
9	WHILE phaseStatusGroupPhaseOns $1 \neq 0x14$ AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	

15.	Set HITL Detector Input 8 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18	WHILE phaseStatusGroupGreens $1 = 0x14$ AND	
101	phaseStatusGroupGreens 2 = 0x00 AND	
	phaseStatusGroupVehCalls 1 AND $0x80 \neq 0x80$	
10	DELAY 1 Second	
20	GET nhaseStatusCrounGreens 1 nhaseStatusGrounGreens 2	
20.	of the phase of a logical of the second to t	
01		
<u> </u>		Dece/Feil
22.	VERIFY phaseStatusGroupGreens.1 = 0x14 AND	Pass/Fall
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 = 0x80	
	Note: Ensure that Phase 8 registers a Vehicle Call during 3+5 Green.	
23.	Set HITL Detector Input 8 = Off	
24.	DELAY .2 Seconds	
Detector 8 calls Ph	hase 8 when Phase 8 and Phase 16 are not Green (In 4+5 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 4 = On	
7	DELAY 2 Seconds	
8	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
0. 0	WHILE phaseStatusGroupPhaseOne 1 $\pm$ 0v18 AND	
Э.	$r_{\text{D}}$	
10	DELAV 1 Second	
10.	CET phono Status Croup Phono One 1	
11.	GET phaseStatusGroupPhaseOns.1,	
40	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HILL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 8 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x18 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 ≠ 0x80	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1. phaseStatusGroupGreens.2.	
	phaseStatusGroupVehCalls.1	
21	WEND	
22	VERIEY phaseStatusGroupGreens 1 = 0x18 AND	Pass/Fail
<i>LL</i> .	nhaseStatusGrounGreens 2 = 0x00 AND	1 455/1 41
	phaseStatusGroupVehCalls 1 AND 0x80 = 0x80	

	Note: Ensure that Phase 8 registers a Vehicle Call during 4+5 Green.	
23.	Set HITL Detector Input 8 = Off	
24.	DELAY .2 Seconds	
Detector 8 calls Ph	nase 8 when Phase 8 and Phase 16 are not Green (In 6+12 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
-	Note: Loop until controller rests in green somewhere.	
<u> </u>	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
10	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
10	pnaseStatusGroupPnaseOns.2	
12.	WEND	
	Nata: Mait until controllar reaches 4+5	
12	Note: Walt until controller reaches $4+5$ .	
11	DELAV 2 Seconds	
14.	Sat HITL Detector Input 6 = On	
16	DELAX 2 Seconds	
10.	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
17.	WHILE nhaseStatusGroupPhaseOns 1 $\pm$ 0x20 AND	
10.	phaseStatusGroupPhaseOns $2 \neq 0x08$	
19	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 6+12.	
22.	Set HITL Detector Input 6 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 8 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x08 AND	
	phaseStatusGroupVenCalls.1 AND 0x80 ≠ 0x80	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
20		
<u> </u>		
31.	vERTET phaseolalusofouporeens. I = 0x20 AND	Pass/Fall
	phaseStatusGrounVehCalle 1 AND 0v80 = 0v80	
	phaseotatasoroup verioalis. 1 AND 0x00 - 0x00	
	Note: Ensure that Phase 8 registers a Vehicle Call during 6+12	
	Green.	
Ц		

32.	Set HITL Detector Input 8 = Off	
33.	DELAY .2 Seconds	
Detector 8 calls Ph	ase 8 when Phase 8 and Phase 16 are not Green (In 6+11 Green)	
1.	GET ringStatus.1. ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3	DELAY 1 Second	
<u></u>	GET ringStatus 1_ringStatus 2	
т. Б		
J.	WEND	
	Noto: Loop until controllor rocts in groop comowhere	
6	Set HITL Detector Input 2 - On	
0.	Set HITL Delector input 3 = On	
1.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DFLAY 2 Seconds	
17	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
18	WHILE nhaceStatusCrounPhaseOne 1 $\pm$ 0x20 AND	
10.	nhaceStatusGrounPhaceOne $2 \pm 0x04$	
10	DELAV 1 Second	
20	CET phaseStatusCroupDhaseOne 1	
20.	DET phaseolalusoloup-haseons.i,	
۷۱.	WEND	
	Noto: Mait until controllor roachas 6+11	
	<b>NOTE:</b> Wall utili controller reduces $0 \neq 11$ .	
<u> </u>		
23.	DELAY 2 Seconds	
24.	Set HILL Detector Input 8 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 ≠ 0x80	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 = 0x80	
	Note: Ensure that Phase 8 registers a Vehicle Call during 6+11	
	Green.	
32.	Set HITL Detector Input 8 = Off	

33.	DELAY .2 Seconds	
Detector 8 calls Ph	ase 8 when Phase 8 and Phase 16 are not Green (In 5+9 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4	GET ringStatus 1 ringStatus 2	
5	WEND	
0.	WEND	
	Note: Loop until controller rests in green somewhere	
6	Set HITL Detector Input $2 = \Omega n$	
7	DELAV 2 Seconds	
γ. 	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
0.	$MUII E nhaceStatusCrounDhaceOne 1 \pm 0x12 AND$	
J.	while phasesialussiouprilaseons $1 \neq 0 \times 12 = 0$	l
10	DELAV 1 Second	
11	DELAT I Second	
11.	GET pridsesialusGroupPhaseOns.1,	1
10	มาการการการการการการการการการการการการการ	
۱∠.	WEND	ĺ
l l	Note Mait until controller reaches 2+5	1
10	Note: Wall until controller reaches 2+5.	
13.	Set HILL Detector input $2 = Off$	
14.	DELAY .2 Seconds	
15.	Set HILL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x10 AND	l
	phaseStatusGroupPhaseOns.2 ≠ 0x01	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	l
	phaseStatusGroupPhaseOns.2	
21.	WEND	l
		l
	Note: Wait until controller reaches 5+9.	
22.	Set HILL Detector Input 6 = Off	
23.	DELAY 2 Seconds	
24.	Set HITL Detector Input 8 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	l
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x10 AND	1
l	phaseStatusGroupGreens.2 = 0x01 AND	1
	phaseStatusGroupVehCalls.1 AND 0x80 ≠ 0x80	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	l
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x10 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x01 AND	l
l l	phaseStatusGroupVehCalls.1 AND 0x80 = 0x80	1
ļ		1
	Note: Ensure that Phase 8 registers a Vehicle Call during 5+9 Green.	
32.	Set HITL Detector Input 8 = Off	
33.	DELAY .2 Seconds	1

Detector 8 extends	s Phase 8 during Phase 8	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns 1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10	DFLAY 1 Second	
11	GET phaseStatusGroupPhaseOns 1	
	phaseStatusGroupPhaseOns 2	
12	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 7 = On	
14	DELAY 2 Seconds	
15	GET ringStatus 2	
10.	WHILE ringStatus 2 AND $0x10 \neq 0x10$ (xxx1xxxx = maxout)	
10.	DELAV 2 Second	
10	CET ringStatus 2	
10.		
19.	WEND	
	Note: Wait for Max Out Indication	
	Ding $1 - 2 - 3 - 4 - 0 - 11 - 12 - 8 - 1$	
	Ring $2 = 15$ 16 5 6 7 8 & 13	
20	GET nhaseStatusGrounPhaseOns 1	
20.	VEDIEV that DESDONSE VALUE phaseStatusCroupDhaseOne 1	Dass/Fail
۷۱.		r ass/1 all
	Note: Ensure that Max Out occurred on Phase 8	
22	Set HITL Detector Input 8 = Off and 7 = Off	
23	DELAY 2 Seconds	
Detector 8 extends	S Phase 16 during Phase 16	
1.	GET ringStatus.1. ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1. ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns 1. phaseStatusGroupPhaseOns 2	
	WHILE phaseStatusGroupPhaseOns $1 \neq 0x81$ AND	
υ.	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11	GET phaseStatusGroupPhaseOns 1	
	nhaseStatusGrounPhaseOns 2	
L		

12.	WEND			
	No (a Mait until controllor receber 4 10			
10	Note: Walt until controller reaches 1+8.			
13.	Set HILL Detector Input 8 = Off			
14.	DELAY .2 Seconds			
15.	Set HILL Detector input 2 = On			
16.	DELAY .2 Seconds			
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2			
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND			
10	DELAY 1 Second			
19. 20	CET phono Status Croup Phono One 1			
20.	GET pridseSidiusGroupPridseOns.1, phaseStatusGroupPhaseOns.2			
21	WEND			
۷۱.	WEND			
	Note: Wait until controller reaches 2+16			
22	Set HITL Detector Input $2 = Off$			
23	DELAY 2 Seconds			
20.	Set HITL Detector Input 8 = On			
25	DELAY 2 Seconds			
26	Set HITL Detector Input 3 = On			
20.	DELAY 2 Seconds			
27.	GET ringStatus 2			
20.	WHILE ringStatus 2 AND $0x10 \neq 0x10$ (xxx1xxxx - maxout)			
<u> </u>	DELAV 1 Second			
31	CET ringStatus 2			
32	WEND			
52.	WEND			
	Note: Wait for Max Out Indication			
	Ring 1 = 2 3 4 9 11 12 & 1			
	Ring $2 = 15$ 16 5 6 7 8 & 13			
33.	GET phaseStatusGroupPhaseOns.2			
34.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2	Pass/Fail		
01.	AND $0x40 = 0x40$			
	Note: Ensure that Max Out occurred on Phase 16.			
35.	Set HITL Detector Input 8 = Off and 3 = Off			
36.	DELAY .2 Seconds			
Teardown	<u> </u>			
	PERFORM Detector Teardown – TC020 if not proceeding to another			
	detector operation test case			
	Test Case Results			
	Date			
Tested By:	Tested			
Test Case Notes:				
Version Historv:	v1.00 05/08/06 Initial Draft – RDR			
	v1.01 07/05/06 Updated notes – RDR			
	v1.02 07/21/06 Implemented script and proofed – JJ			

## Detector 9 Operations

Test Case:	Title:	Detector 9 Operations	
TC009	Description: Verifies the operation of Detector 9 to call Phase 6 under		
	specific conditions, extend Phase 2 under specific conditions.		
		and extend intervals 2516B, 2517B, 2518B, 3517B	. 3518B.
		4517B. and 4518B.	,,
	Constants:		
	Variables:		
	Pass/Fail	The DUT shall pass every verification step included	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Sten	Test Procedure		Results
Number	1031110000010		Nosuns
Setup			
Cottap	PERFORM Detecto	or Operations Setup – TC019 if not already done	
	SO		
Detector 9 No Call	on Phase 6 when C	)verlan A is Green	
1	GET ringStatus 1	ringStatus 2	
2	WHILE ringStatus	$1 \neq 0x03$ AND ringStatus 2 $\neq 0x03$	
<u>ک.</u> ۲	DELAV 1 Secon	$\tau \neq 0.000 \text{ Alve migotatas.} \neq 0.000$	
<u> </u>	GET ringStatus	1 ringStatus 2	
<u>т.</u> Б			
5.	VVEND		
	Noto: Loop until co	ntrollor rosts somowhoro	
6	Sot UITL Detector		
0.	DELAX 2 Seconda		
/.	DELAT .2 Seconds		
Ŏ.			
9.	WHILE phaseStatu	ISGroupPhaseOns.1 ≠ UX21 AND	
10	pnaseStatusGroup	PhaseOns.2 ≠ UXUU	
10.	DELAY 1 Secor		
11.	GET phaseStat	usGroupPhaseOns.1,	
	phaseStatusGro	pupPhaseOns.2	
12.			
10	Note: Wait until cor	itroller reaches 1+6.	
13.	Set HIL Detector	input 6 = $O \pi$	
14.	DELAY .2 Seconds		
15.	GET ringStatus.1, I		
16.	WHILE ringStatus.	$1 \neq 0x03$ AND ringStatus.2 $\neq 0x03$	
17.	DELAY 1 Secor		
18.	GET ringStatus	.1, ringStatus.2	
19.	WEND		
	Note: Wait for 1+6	Green Rest.	
20.	GET overlapStatus	GroupGreens.1	Pass/Fail
	Note: overlapStatus	sGroupGreens is optional and a GET may return	
	a noSuchName. T	his also assumes that Overlap $A = 1+2+X$ .	
21.	IF RESPONSE ER	ROR = noError THEN	
22.	VERIFY that RE	SPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND $0x01 = 0x0$	01	
	Note: Verifies that	Overlap A = Green.	

23.	ENDIF	
24.	Set HITL Detector Input 9 = On	
25.	DELAY 2 Seconds (2 full seconds)	
26	GET phaseStatusGroupVehCalls 1 phaseStatusGroupVehCalls 2	
27	VERIEV that RESPONSE VALUE phaseStatusGroupVehCalls 1 =	Pass/Fail
27.	0x00  AND phaseStatusGroupVehCalls  = 0x00	
	Note: Verify that controller does not register call	
28	Set HITL Detector Input $9 = Off$	
29	DELAY 2 Seconds	
Detector 9 No Call	on Phase 6 when Overlan A is not Green and a Call on Phase 7	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03	
<u>ک</u> .	DELAY 1 Second	
<u></u> Л	CET ringStatus 1 ringStatus 2	
<u>т.</u> Б		
5.	WEND	
	Note: Loop until controller rests somewhere	
6	Set HITI Detector Input $4 = \Omega n$	
7	DELAY 2 Seconds	
<u> </u>	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
0.	MHII = phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	while phaseStatusGloupFlaseOlis.1 $\neq$ 0x10 AND phaseStatusCroupDhaseOne 2 $\neq$ 0X00	
10	DELAV 1 Second	
10.	DELAT I Second	
11.	GET phaseStatusGroupPhaseOns. 1,	
10		
12.	WEIND	
10	Note: Wall until controller reaches 4+5.	
13.	Set HILL Detector input $4 = Off$	
14.	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	Pass/Fall
	Notes everlar Status Crown Crowns is entional and a CET may return	
	Note: overlapStatusGroupGreens is optional and a GET may return	
40	a nosuchivame. This also assumes that Overlap $A = 1+2+X$ .	
10.		D/E-1
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fall
	AND $0x01 = 0x00$	
	$N_{ata}$ , $V_{arifica}$ that $\Omega_{Varian} \Lambda = NOT$ Organ	
10	Note. Verifies that Overlap A = NOT Green	
10.	ENDIF Set LUTI Detector Innut 7 - On	
19.	Set HITL Detector input 7 = On	
20.	DELAY .2 Seconds	
21.	Set HILL Detector input and 9 = On	
22.	DELAY 2 Seconds (2 full seconds)	
23.		Pass/Fail
	AND $0x20 = 0$	
	Nata: Chaska far na sall an Dhasa C	
24.		
<u></u> 25.	Set HIL Detector input $9 = OT and 7 = OT$	
26.	DELAY 2 Seconds	
Detector 9 No Call	on Phase 6 when Overlap A is not Green and a Call on Phase 8	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	

3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	Pass/Fail
	<i>Note:</i> overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap $A = 1+2+X$ .	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND 0x01 = 0x00	
	<i>Note:</i> Verifies that Overlap A = NOT Green	
18.	ENDIF	
19.	Set HITL Detector Input 8 = On	
20.	DELAY .2 Seconds	
21.	Set HITL Detector Input 9 = On	
22.	DELAY 2 Seconds (2 full seconds)	
23.	GET phaseStatusGroupVehCalls.1	
24.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1	Pass/Fail
	AND $0x20 = 0$	
	Note: Checks for no call on Phase 6	
25.	Set HITL Detector Input 9 = Off and 8 = Off	
26.	DELAY .2 Seconds	
Detector 9 Calls Pl	nase 6 when Overlap A is not Green and no Calls on Phase 7 or Pha	ise 8
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	

12	WEND	
	Note: Wait until controller reaches 3+5	
1.3	Set HITL Detector Input 3 = Off	
10.	DELAY 2 Seconds	
15. 15	GET overlanStatusGrounGreens 1	Dass/Fail
15.	GET ovenapstatus Group Greens. T	F 855/F 811
	Note: overlanStatusGrounGreens is ontional and a CET may return	
	Note: Overlap Status Group Greens is optional and a GET may return a negative Name. This also assumes that Overlap $\Lambda = 1+2+Y$	
16	a nosucinvalue. This also assumes that Overlap $A = 1+2+\Lambda$ .	
10.	IF RESPONSE ERROR - INCLUE IN ITEN	
17.		Fass/Fall
	AND 0x01 = 0x00	
	Note: Verifies that Overlap $\Lambda = NOT$ Green	
10		
10.	ENDIF Set HITL Detector Input 0 = On	
19.	Set HITL Detector input 9 = On	
20.	DELAY 2 Seconds (2 full seconds)	
21.		
22.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1	Pass/Fail
	AND $0x20 = 0x20$	
	Note: Checks for call on Phase 6	
23.	Set HIIL Detector Input 9 = Off	
24.	DELAY .2 Seconds	
Detector 9 does no	ot Extends Phase 2 when Phase 2 is Green and there are no Calls o	n Phase 3 or
Phase 4		
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 2 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 9 = On	
	Note: To possible extend	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 2 = On	
	Note: To force extensions	
18.	DELAY .2 Seconds	

19	Set HITL Detector Input 8 = On	
10.		
	Note: When opposing call exists	
20	DELAY 2 Seconds	
21	Set HITL Detector Input 2 = Off	
<b>2</b> · · ·		
	Note: To check whether possible extensions are due to Detector 9	
22	DELAY 2 Seconds	
23	GET ringStatus 1	
20.	WHILE ringStatus 1 AND $0x08 \neq 0x08$ (xxxx1xxx = gap out)	
25	DELAY 1 Second	
26	GET ringStatus 1 ringStatus 2	
20.	WEND	
21.	WEND	
	Note: Wait for Gan Out indication on Ring 1	
	Ring $1 = 2$ 3 4 9 11 12 & 1	
	Ring $2 = 15$ 16 5 6 7 8 & 13	
28	GET phaseStatusGroupPhaseOns 1	
20.	VERIEV that RESPONSE VALUE nhaseStatusGrounPhaseOns 1	Pass/Fail
20.	AND 0x02 = 0x02	1 000/1 01
	Note: Verify that Gap Out indication occurred on Phase 2.	
30	Set HITL Detector Input 8 and 9 = $Off$	
31	DELAY 2 Seconds	
Detector 9 extends	S Phase 2 when Phase 2 is Green and a Call on Phase 3	
1.	GET ringStatus.1. ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3.	DELAY 1 Second	
4	GET ringStatus 1, ringStatus 2	
5	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 2 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 9 = On	
	Note: To possible extend	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 2 = On	
	Note: To force extensions	
18.	DELAY .2 Seconds	

19	Set HITL Detector Input 3 = On	
10.		
	Note: When opposing call exists	
20	DELAV 2 Seconds	
20.	Set HITL Detector Input 2 - Off	
21.		
	Note: To check whether possible extensions are due to Detector 9	
<u>ົ</u>	DELAX 2 Secondo	
<u> </u>	CET ringStatue 1	
23. 24	GET IIIgStatus. I WEII E ringStatus 1 AND $0x10 \neq 0x10$ (voor10000 = moveut)	
24. 25	$\frac{1}{2} = \frac{1}{2} = \frac{1}$	
<u> </u>	DELAT I SECOLU	
20.		
۷۱.	WEND	
	Note: Weit for May Out Indication on Ding 1	
	$P_{\text{ind}} = \frac{1}{2} + \frac$	
	$\begin{array}{c} \text{Ring 1} = \mathbf{Z},  3,  4,  9,  11,  12,  \mathbf{\alpha} \\ \text{Ding 2} = 15,  16,  5,  6,  7,  9,  212 \end{array}$	
20	CET phaseStatusCroupDhaseOne 1	
20.	VEDIEV that DESDONSE VALUE phaseStatusCroupDhaseOne 1	Deco/Eail
29.		Pass/Fall
	AND $0x0z = 0x0z$	
	Noto: Varify that Max Out indication occurred on Phase 2	
20	Note: Verify that was Out indication occurred on Filase 2 Set $HITL$ Detector input 2 and 0 = Off	
21	DELAX 2 Secondo	
Detector 9 extends	Phase 2 when Phase 2 is Green and a Call on Phase 4	
	GET ringStatus 1 ringStatus 2	
1. 2	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
2.	DELAV 1 Second	
<u>J</u>	GET ringStatus 1, ringStatus 2	
<u> </u>		
5.	WEND	
	Note: Loop until controller rests somewhere	
6	Set HITL Detector Input 2 - On	
7	DELAV 2 Seconds	
<u> </u>	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
0.	WHII = phaseStatusGloupFildseOffs.1, phaseStatusGloupFildseOffs.2	
9.	nhaseStatusCrounPhaseOns 2 + 0X00	
10	DELAV 1 Second	
10.	CET phaseStatusGroupPhaseOns 1	
11.	obaseStatusGroupPhaseOns 2	
12		
12.	Note: Wait until controller reaches 2+5	
13	Set HITL Detector Input $2 = Off$	
10.	DELAV 2 Seconds	
15	Set HITL Detector Input 9 - On	
13.		
	Noto: To possible extend	
16	DELAV 2 Seconde	
10.	Sat HITL Detector Input 2 - On	
17.		
	Note: To force extensions	
18	DELAY 2 Seconds	
10.		

19.	Set HITL Detector Input 4 = On	
	<b>.</b>	
	Note: When opposing call exists	
20.	DELAY 2 Seconds	
21.	Set HITL Detector Input 2 = Off	
	Noto: To shook whether possible extensions are due to Detector 0	
20	DELAX 2 Seconda	
<u> </u>	CET ringStatus 1	
23.	$\frac{\text{OET IIIIgotatus.T}}{\text{WHILE ringStatus 1 AND 0x10 + 0x10 (xxx1xxxx - maxout)}}$	
24.	$\frac{1}{10000000000000000000000000000000000$	
25.	CET ringStatus 1	
20.	WEND	
21.	WEND	
	Note Wait for Max Out Indication on Ring 1	
	Ring $1 = 2$ 3 4 9 11 12 & 1	
	Ring $2 = 15$ 16 5 6 7 8 & 13	
28	GET nhaseStatusGrounPhaseOns 1	
20.	VEDIEV that DESDONSE VALUE phaseStatusCroupDhaseOns 1	Dass/Fail
29.		r ass/r all
	AND 0x02 - 0x02	
	Note: Verify that Max Out indication occurred on Phase 2	
30	Set HITL Detector Input 4 and 9 = Off	
	DELAY 2 Seconds	
Detector 9 extends	Phase 9 when Phase 9 is Green	
	CET ringStatus 1 ringStatus 2	
۱. ر	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
2.	DELAV 1 Second	
J.	CET ringStatue 1 ringStatue 2	
4. 5		
5.	WEND	
	Note: Loop until controller rests somewhere	
6	Set HITL Detector Input 4 = On	
7	DELAY 2 Seconds	
<u> </u>	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
0.	WHILE nhaseStatusGroupPhaseOns 1 $\pm$ 0v18 AND	
Э.	$r_{\text{mase}}$	
10	DELAV 1 Second	
11	CET nhaseStatusCrounPhaseOns 1	
11.	nhaseStatusGrounPhaseOns 2	
12		
12.	Note: Wait until controller reaches 4+5	
13	Set HITL Detector Input $4 = Off$	
10.	DELAV 2 Seconds	
15	Set HITL Detector Input 7 - On	
16	DELAX 2 Seconde	
17	Set HITL Detector Input 0 = On	
17.	DELAV 2 Secondo	
10.	DELAT .2 Seconds	
19.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
∠0.	while phaseStatusGroupPhaseOns.1 ≠ 0X10 AND	
	priaseStatusGroupPriaseOns.2 ≠ UXU1	
21.	DELAT I Second	
22.	GEI phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	

23.	WEND			
	Note: Wait until controller reaches 5+9.			
24.	Set HITL Detector Input 7 = Off			
25.	DELAY .2 Seconds			
26.	GET ringStatus.1			
27.	WHILE ringStatus.1 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)			
28.	DELAY 1 Second			
29.	GET ringStatus.1			
30.	WEND			
	Note: Wait for Max Out Indication.			
	Ring 1 = 2, 3, 4, <b>9</b> , 11, 12, & 1			
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13			
31.	GET phaseStatusGroupPhaseOns.2			
32.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2	Pass/Fail		
	AND 0x01 = 0x01			
	Note: Ensure that Max Out occurred on Phase 9.			
33.	Set HITL Detector Input 9 = Off			
34.	DELAY .2 Seconds			
Teardown				
	PERFORM Detector Teardown – TC020 if not proceeding to another			
	detector operation test case			
	Test Case Results			
Tootod Dy:	Date	Doog/Eail		
Testeu by.	Tested	Fass/Fall		
Test Case Notes:	<notes></notes>			
Version History:	v1.00 05/09/06 Initial Draft – RDR			
	v1.01 07/05/06 Updated notes – RDR			
	v1.02 07/17/06 Implemented script and proofed – JJ			

## Detector 10 Operations

Test Case:	Title:	Detector 10 Operations	
TC010	Description:	Verifies the operation of Detector 10 to call Phase 6 under specific conditions, extend Phase 2 under specific conditions, and extend intervals 2516B, 2517B, 2518B, 3517B, 3518B, 4517B, and 4518B.	
	Constants:		
	Variables:		
	Pass/Fail	The DUT shall pass every verification step include	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detect	or Operations Setup – TC019 if not already done	
	SO.		
Detector 10 No Ca	II on Phase 6 when	Overlap A is Green	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus.	1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Seco	nd	

4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8	GET phaseStatusGroupPhaseOns 1 phaseStatusGroupPhaseOns 2	
9	WHILE phaseStatusGroupPhaseOns $1 \neq 0x21$ AND	
0.	nhaseStatusGroupPhaseOns $2 \neq 0x00$	
10	DELAY 1 Second	
11	GET nhaseStatusGrounPhaseOns 1	
	nhaseStatusGrounPhaseOns 2	
12	WEND	
12.	WEND	
	Note: Wait until controller reaches 1+6	
10	Note. Walt until controller reaches 1+0.	
ای. ۱۵.	DELAV 2 Secondo	
14.	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
17.	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Wait for 1+6 Green Rest.	
20.	GET overlapStatusGroupGreens.1	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap A = 1+2+X.	
21.	IF RESPONSE ERROR = noError THEN	
22.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND 0x01 = 0x01	
	<i>Note:</i> Verifies that Overlap A = Green	
23.	END IF	
24.	Set HITL Detector Input 10 = On	
25.	DELAY 2 Seconds	
26.	GET phaseStatusGroupVehCalls 1, phaseStatusGroupVehCalls 2	
27	VERIEV that RESPONSE VALUE phaseStatusGroupVehCalls 1 =	Pass/Fail
	0x00  AND phaseStatusGroupVehCalls  = 0x00	
	Note: Verify that controller does not register call	
28	Set HITL Detector Input 10 = Off	
20.	DELAY 2 Seconds	
Detector 10 No Ca	II on Phase 6 when Overlan A is not Green and a Call on Phase 7	
	GET ringStatus 1 ringStatus 2	
່. ວ	WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03	
<u>∠.</u>	DELAV 4 Second	
J. 1	DELAT I Secoliu	
4. F		
5.	WEND	
	Notes Lean until controller reate construit are	
	Note: Loop until controller rests somewhere.	
<u>6</u> .	Set HILL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	

9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	
	Note: everlap Status Crown Crosses is entional and a CET may return	
	Note: overlapStatusGroupGreens is optional and a GET may return	
16	a nosuchivalitie. This also assumes that Overlap $A = 1+2+X$	
10.		Dece/Eeil
17.		Pass/Fall
	AND $0x01 = 0x00$	
	Note: Verifies that Overlap $A = NOT$ Green	
18	FND IF	
19	Set HITL Detector Input 7 = On	
20	DELAY 2 Seconds	
21	Set HITL Detector Input 10 = On	
22	DELAY 2 Seconds	
23	GET phaseStatusGroup\/ebCalls 1	
20.	VERIEV that RESPONSE VALUE nhaseStatusGrounVehCalls 1	Pass/Fail
۲.	AND $0x20 = 0$	1 435/1 41
	Note: Checks for no call on Phase 6	
25.	Set HITL Detector Input 10 = Off and 7 = Off	
26.	DELAY .2 Seconds	
Detector 10 No Ca	II on Phase 6 when Overlap A is not Green and a Call on Phase 8	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note Moit until controller reaches 4:5	
10	Ivote: vvali until controller reaches 4+5.	
13.		
14.	DELAT .2 Seconds	

15.	GET overlapStatusGroupGreens.1	
	Note: overlapStatusGroupGreens is optional and a GET may return	
4.0	a noSuchName. This also assumes that Overlap A = 1+2+X	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND $0x01 = 0x00$	
	$\lambda$ (ata) (arifica that $\Omega$ ) (arian $\Lambda = NOT$ Organ	
10	Note: veniles that Ovenap A = NOT Green.	
10.	END IF	
19.	DELAV 2 Seconda	
20.	DELAT .2 Seconds	
21. 22	DELAX 2 Secondo	
22.	DELAT 2 Seconds	
<u> </u>	GET phaseStatusGroupVenCalls. T	Dece/Fail
24.		Pass/Fail
	Note: Checks for no call on Phase 6	
25	Set HITL Detector Input $10 = Off$ and $8 = Off$	
20.	DELAY 2 Seconds	
Detector 10 Calls I	Phase 6 when Overlan A is not Green and no Calls on Phase 7 or Ph	1250 8
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
3	DELAY 1 Second	
0.	GET ringStatus 1 ringStatus 2	
5	WEND	
0.		
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap A = 1+2+X	
16.		D / 5 ''
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	$N_{oto}$ $V_{orifice that Overlap A = NOT Orean$	
10	Note, veniles that Ovenap A = NOT Green.	
10.	Sat HITL Detector Input 10 - On	
າອ. 	DELAY 2 Seconds	
<u> </u>		

21.	GET phaseStatusGroupVehCalls.1	
22.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1	Pass/Fail
	AND 0x20 = 0x20	
	Note: Checks for call on Phase 6	
23.	Set HITL Detector Input 10 = Off	
24	DELAY 2 Seconds	
Detector 10 does n	not Extends Phase 2 when Phase 2 is Green and there are no Calls	on Phase 3
or Phase 4		
1	GET ringStatus 1 ringStatus 2	
۱. ۲	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
2. 2	DELAV 1 Second	
<u>.</u>	CET ringStatus 1 ringStatus 2	
Э.	WEND	
	Note Lean until controller reste computers	
6	Note. Loop unui controller rests somewhere.	
0.	Set $\Pi \Pi L$ Detector input $2 = On$	
<u> </u>	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 10 = On	
	Note: To possible extend	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 2 = On	
	Note: To force extensions	
18.	DELAY .2 Seconds	
19.	Set HITL Detector Input 8 = On	
	Note: When opposing call exists	
20.	DELAY .2 Seconds	
21.	Set HITL Detector Input 2 = Off	
	<i>Note:</i> To check whether possible extensions are due to Detector 10.	
22.	DELAY .2 Seconds	
23.	GET ringStatus.1	
24.	WHILE ringStatus 1 AND $0x08 \neq 0x08$ (xxxx1xxx = gap out)	
25	DELAY 1 Second	
26	GET ringStatus 1 ringStatus 2	
20.	WEND	
۷۱.		
	Note: Wait for Gan Out indication on Ring 1	
	Ping 1 = $2.3 \text{ / } 0.11  12.8  1$	
	Ding $2 - 15$ 16 5 6 7 8 8 13	
	$1 \times 10^{-10} = 10^{-10}, 10^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^{-0}, 0^$	

28.	GET phaseStatusGroupPhaseOns.1	
29.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x02 = 0x02	
	Note: Verify that Gap Out indication occurred on Phase 2	
30.	Set HITL Detector Input 8 and 9 = Off	
31	DELAY 2 Seconds	
Detector 10 extend	Is Phase 2 when Phase 2 is Green and a Call on Phase 3	
1	GET ringStatus 1 ringStatus 2	
ן. ס	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
2.	DELAV 1 Second	
J.	CET ringStatus 1 ringStatus 2	
4. F		
Э.	WEND	
	No (a di anno suntil angeter llan na ta angeter angeter a	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 2 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 10 = On	
	<i>Note:</i> To possible extend	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 2 = On	
	Note: To force extensions	
18	DELAY 2 Seconds	
19	Set HITL Detector Input 3 = On	
10.		
	Note: When opposing call exists	
20	DELAX 2 Seconds	
20.	Set HITL Detector Input 2 - Off	
21.		
	Note: To check whether possible extensions are due to Detector 10	
22	DELAX 2 Seconde	
22.	CET ring Status 1	
<u>کې</u>	GET IIIgStatus. I M/H = rigs Otatus (1 AND 0.110 + 0.110 (mm.1.mm) = resource t)	
24.	WHILE INGStatus. I AND $0x10 \neq 0x10$ (xxx1xxxx = maxout)	
25.	DELAT 1 Second	
26.	GET ringStatus.1	
27.	WEND	
	Note: Wait for Max Out Indication on Ring 1	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
28.	GET phaseStatusGroupPhaseOns.1	

29.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x02 = 0x02	
	Note: Verify that Max Out indication occurred on Phase 2	
30.	Set HITL Detector Input 3 and 9 = Off	
31.	DELAY .2 Seconds	
Detector 10 extend	Is Phase 2 when Phase 2 is Green and a Call on Phase 4	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note Lean until controller reste computers	
~	Note: Loop until controller rests somewhere.	
0.	Set HILL Detector Input 2 = On	
1.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
10	pnaseStatusGroupPnaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
10		
12.	WEND	
	Noto: Wait until controller reaches 2+5	
13	Set HITL Detector Input 2 - Off	
10.	DELAV 2 Seconds	
14.	Set HITL Detector Input 10 - On	
15.		
	Note: To possible extend	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 2 = On	
	Note: To force extensions	
18.	DELAY .2 Seconds	
19.	Set HITL Detector Input 4 = On	
	Note: When opposing call exists	
20.	DELAY .2 Seconds	
21.	Set HITL Detector Input 2 = Off	
~~~	Note: To check whether possible extensions are due to Detector 10	
22.	DELAY .2 Seconds	
23.	GET ringStatus.1	
24.	WHILE ringStatus.1 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
25.	DELAY 1 Second	
26.	GET ringStatus.1	
27.	WEND	
	Note: Weit for May Out Indication on Ding 1	
	Prote, wait for wax Out indication on King 1 Ding $1 - 2 - 3 - 4 - 0 - 11 - 12 - 8 - 1$	
	$\begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 $	
റം	$\frac{1}{10} = 10, 10, 0, 0, 1, 0, 0 = 1$	
20.	GLI phaseolalusolouprilaseolis. I	

29.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	Note: Verify that Max Out indication occurred on Phase 2	
30.	Set HITL Detector Input 4 and 9 = Off	
31.	DELAY .2 Seconds	
Detector 10 extend	Is Phase 9 when Phase 9 is Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus 1 ≠ 0x03 AND ringStatus 2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
40	pnaseStatusGroupPnaseOns.2	
12.	WEND	
	Noto: Wait until controllor roaches 4+5	
13	Note. Walt until controller reaches $4+5$.	
11.	DELAY 2 Seconds	
14.	Set HITL Detector Input $7 = \Omega n$	
16	DELAY 2 Seconds	
10.	Set HITL Detector Input 10 = On	
17.	DELAY 2 Seconds	
19	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
20.	WHILE phaseStatusGroupPhaseOns $1 \neq 0x10$ AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x01	
21.	DELAY 1 Second	
22.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
23.	WEND	
-	Note: Wait until controller reaches 5+9.	
24.	Set HITL Detector Input 7 = Off	
25.	DELAY .2 Seconds	
26.	GET ringStatus.1	
27.	WHILE ringStatus.1 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
28.	DELAY 1 Second	
29.		
30.		
	Note: Wait for Max Out Indication	
	Ring $1 = 2 3 4 9 11 12 \& 1$	
	Ring $2 = 15$ 16 5 6 7 8 & 13	
31.	GET phaseStatusGroupPhaseOns.2	
u		

32.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2	Pass/Faii
	AND 0x01 = 0x01	
	Note: Ensure that Max Out occurred on Phase 9	
	A children and the state of the	
33.	Set HITL Detector Input 10 = Off	
34.	DELAY .2 Seconds	
Teardown		
	PERFORM Detector Teardown – TC020 if not proceeding to another	
	detector operation test case	
	Test Case Results	
Tested D.	Date	
Tested By:	Tested	
Test Case Notes:		
Version History:	v1.00 05/09/06 Initial Draft – RDR	
-	v1.01.07/05/06 Updated notes – RDR	
	v1.02 07/19/06 Implemented script and prooted – JJ	

Detector 11 Operations

Test Case: Thic. Detector 11 Operations Description: Verifies the operation of Detector 11 to call and extend Phase 2 under specific conditions Constants: Variables: currentExtendValue Pass/Fail The DUT shall pass every verification step included within the Criteria: Test Case. Test Step Test Procedure Results Number PERFORM Detector Operations Setup – TC019 if not already done so. so. Setup Setup Employed and the system of the system o	Toot Coool	Titlo	Detector 11 Operations	
TCUT1 Description: Vermes the operation of Detector 11 to Call and extend Phase 2 under specific conditions Constants: Variables: currentExtendValue Pass/Fail The DUT shall pass every verification step included within the Criteria: Test Case Test Step Test Procedure Results Number PERFORM Detector Operations Setup – TC019 if not already done so. Results Detector 11 calls Phase 2 when Phase 2 not Green (In 3+5 Green) Employee 1. GET ringStatus.1, ringStatus.2			Verifies the exercise of Detector 11 to cell and out	
under specific conditionsConstants: Variables: currentExtendValue Pass/FailThe DUT shall pass every verification step included within the Criteria: Test Case in order to pass the Test Case.Test Step NumberTest ProcedureResultsSetupPERFORM Detector Operations Setup – TC019 if not already done so.Detector 11 calls Phase 2 when Phase 2 not Green (In 3+5 Green)1.GET ringStatus.1, ringStatus.22.WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x033.DELAY 1 Second4.GET ringStatus.1, ringStatus.25.WENDNote: Loop until controller rests somewhere.6.Set HITL Detector Input 3 = On7.DELAY 2 Seconds8.GET phaseStatusGroupPhaseOns.1 \neq 0x14 AND phaseStatusGroupPhaseOns.2 \neq 0x0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1 \neq 0x14 AND phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 \neq 0x14 AND phaseStatusGroupPhaseOns.211.GET phaseStatusGroupPhaseOns.1 \neq 0x14 AND phaseStatusGroupPhaseOns.212.WENDNote: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY 2 Seconds15.Set HITL Detector Input 11 = On	10011	Description:	verifies the operation of Detector 11 to call and ext	end Phase 2
Constants: Variables: currentExtendValue Pass/Fail The DUT shall pass every verification step included within the Criteria: Test Case in order to pass the Test Case. Test Step Test Procedure Results Number PERFORM Detector Operations Setup – TC019 if not already done so. Image: Constants: Setup PERFORM Detector Operations Setup – TC019 if not already done so. Image: Constants: Detector 11 calls Phase 2 when Phase 2 not Green (In 3+5 Green) Image: Constants: Image: Constants: 1. GET ringStatus.1, ringStatus.2 Image: Constants: Image: Constants: 2. WHILE ringStatus.1 # 0x03 AND ringStatus.2 # 0x03 Image: Constants: Image: Constants: 3. DELAY 1 Second Image: Constants: Image: Constants: Image: Constants: 5. WEND Image: Constants: Image: Constants: Image: Constants: Image: Constants: 6. Set HITL Detector Input 3 = On Image: Constants: Image: Constants: </td <td></td> <td></td> <td>under specific conditions</td> <td></td>			under specific conditions	
Variables: currentExtendValue Pass/Fail The DUT shall pass every verification step included within the Criteria: Test Step Test Procedure Number Results Setup PERFORM Detector Operations Setup – TC019 if not already done so. 0 GET ringStatus.1, ringStatus.2 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests somewhere. 6. Set HITL Detector Input 3 = On 7. DELAY 2 Seconds 8. GET phaseStatusGroupPhaseOns.1, ≠ 0x14 AND phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1, ± 0x14 AND phaseStatusGroupPhaseOns.2 MEND 11. GET phaseStatusGroupPhaseOns.1, ± 0x14 AND phaseStatusGroupPhaseOns.2 MEND 11. GET phaseStatusGroupPhaseOns.1, ± 0x14 AND phaseStatusGroupPhaseOns.2 MEND 12. WEND 13. Set		Constants:		
Pass/Fail The DUT shall pass every verification step included within the Criteria: Test Case in order to pass the Test Case. Test Step Test Procedure Results Number PERFORM Detector Operations Setup – TC019 if not already done so. Results Detector 11 calls Phase 2 when Phase 2 not Green (In 3+5 Green) Image: Comparison of the state of th		Variables:	currentExtendValue	
Criteria: Test Case in order to pass the Test Case. Test Step Number Test Procedure Results Setup PERFORM Detector Operations Setup – TC019 if not already done so. Results Detector 11 calls Phase 2 when Phase 2 not Green (In 3+5 Green) Image: Comparison of the solution of the solutis of the solution of		Pass/Fail	The DUT shall pass every verification step included	d within the
Test Step Number Test Procedure Results Setup PERFORM Detector Operations Setup – TC019 if not already done so. Interface Detector 11 calls Phase 2 when Phase 2 not Green (In 3+5 Green) Interface 1. GET ringStatus.1, ringStatus.2 Interface 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 Interface 3. DELAY 1 Second Interface 4. GET ringStatus.1, ringStatus.2 Interface 5. WEND Interface 0. Note: Loop until controller rests somewhere. Interface 6. Set HTL Detector Input 3 = On Interface 7. DELAY .2 Seconds Interface 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 Interface 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 Interface 11. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 Interface 12. WEND Note: Wait until controller reaches 3+5. Interface 13. Set HITL Detector Input 3 = Off Interface 14. DELAY .2 Seconds Interface 15. Set HITL Detector Input 11		Criteria:	Test Case in order to pass the Test Case.	
Number PERFORM Detector Operations Setup – TC019 if not already done so. Setup so. Detector 11 calls Phase 2 when Phase 2 not Green (in 3+5 Green) 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND 6. Set HITL Detector Input 3 = On 7. DELAY 2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 10. DELAY 1 Second 11. GET phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 11. GET phaseStatusGroupPhaseOns.2 ≠ 0X00 12. WEND Note: Wait until controller reaches 3+5. 13. Set HITL Detector Input 3 = Off 14. DELAY 2 Seconds 15. Set HITL Detector Input 3 = Off	Test Step	Test Procedure		Results
Setup PERFORM Detector Operations Setup – TC019 if not already done so. Detector 11 calls Phase 2 when Phase 2 not Green (In 3+5 Green)	Number			
PERFORM Detector Operations Setup – TC019 if not already done so. Image: Sole Detector 11 calls Phase 2 when Phase 2 not Green (In 3+5 Green) Image: Sole 1. GET ringStatus.1, ringStatus.2 Image: Sole 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 Image: Sole 3. DELAY 1 Second Image: Sole 4. GET ringStatus.1, ringStatus.2 Image: Sole 5. WEND Image: Sole Note: Loop until controller rests somewhere. Image: Sole 6. Set HITL Detector Input 3 = On Image: Sole 7. DELAY .2 Seconds Image: Sole 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 Image: Sole 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 Image: Sole 10. DELAY 1 Second Image: Sole Image: Sole 11. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 Image: Sole Image: Sole 12. WEND Image: Sole Image: Sole Image: Sole Image: Sole 13. Set HITL Detector Input 3 = Off Image: Sole Image: Sole <td< th=""><th>Setup</th><th></th><th></th><th></th></td<>	Setup			
so.Detector 11 calls Phase 2 when Phase 2 not Green (In 3+5 Green)1.GET ringStatus.1, ringStatus.22.WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x033.DELAY 1 Second4.GET ringStatus.1, ringStatus.25.WENDNote: Loop until controller rests somewhere.6.Set HITL Detector Input 3 = On7.DELAY 2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 \neq 0x14 AND10.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.212.WENDNote: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY 1 Second15.Set HITL Detector Input 3 = Off14.DELAY 1 Seconds		PERFORM Detect	or Operations Setup – TC019 if not already done	
Detector 11 calls Phase 2 when Phase 2 not Green (In 3+5 Green)1.GET ringStatus.1, ringStatus.22.WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x033.DELAY 1 Second4.GET ringStatus.1, ringStatus.25.WENDNote: Loop until controller rests somewhere.6.Set HITL Detector Input 3 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 ≠ 0X0012.WEND Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds		SO.		
Detector 11 calls Phase 2 when Phase 2 not Green (In 3+5 Green) 1. GET ringStatus.1, ringStatus.2 2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests somewhere. 6. Set HITL Detector Input 3 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second 11. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 12. WEND Note: Wait until controller reaches 3+5. 13. Set HITL Detector Input 3 = Off 14. DELAY .2 Seconds 15. Set HITL Detector Input 3 = Off				
1.GET ringStatus.1, ringStatus.22.WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x033.DELAY 1 Second4.GET ringStatus.1, ringStatus.25.WENDNote: Loop until controller rests somewhere.6.Set HITL Detector Input 3 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 \neq 0x14 AND phaseStatusGroupPhaseOns.2 \neq 0X0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 1 = On	Detector 11 calls P	hase 2 when Phas	e 2 not Green (In 3+5 Green)	
2. WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests somewhere. 6. Set HITL Detector Input 3 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 10. DELAY 1 Second 11. GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2 12. WEND Note: Wait until controller reaches 3+5. 13. Set HITL Detector Input 3 = Off 14. DELAY .2 Seconds 15. Set HITL Detector Input 11 = On	1.	GET ringStatus.1,	ringStatus.2	
3. DELAY 1 Second 4. GET ringStatus.1, ringStatus.2 5. WEND Note: Loop until controller rests somewhere. 6. Set HITL Detector Input 3 = On 7. DELAY .2 Seconds 8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 10. DELAY 1 Second 11. GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2 12. WEND Note: Wait until controller reaches 3+5. 13. Set HITL Detector Input 3 = Off 14. DELAY .2 Seconds 15. Set HITL Detector Input 11 = On	2.	WHILE ringStatus.	$1 \neq 0x03$ AND ringStatus.2 $\neq 0x03$	
4.GET ringStatus.1, ringStatus.25.WENDNote: Loop until controller rests somewhere.6.Set HITL Detector Input 3 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On	3.	DELAY 1 Seco	nd	
5.WENDNote: Loop until controller rests somewhere.6.Set HITL Detector Input 3 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On	4.	GET ringStatus	.1, ringStatus.2	
Note: Loop until controller rests somewhere.6.Set HITL Detector Input 3 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On	5.	WEND		
Note: Loop until controller rests somewhere.6.Set HITL Detector Input 3 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On				
6.Set HITL Detector Input 3 = On7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On		Note: Loop until co	ntroller rests somewhere.	
7.DELAY .2 Seconds8.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.29.WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X0010.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On	6.	Set HITL Detector	Input 3 = On	
8. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 10. DELAY 1 Second 11. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 12. WEND Note: Wait until controller reaches 3+5. 13. Set HITL Detector Input 3 = Off 14. DELAY .2 Seconds 15. Set HITL Detector Input 11 = On	7.	DELAY .2 Seconds	8	
9. WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 10. DELAY 1 Second 11. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 12. WEND Note: Wait until controller reaches 3+5. 13. Set HITL Detector Input 3 = Off 14. DELAY .2 Seconds 15. Set HITL Detector Input 11 = On	8.	GET phaseStatus	GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
phaseStatusGroupPhaseOns.2 ≠ 0X00 10. DELAY 1 Second 11. GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 12. WEND Note: Wait until controller reaches 3+5. 13. Set HITL Detector Input 3 = Off 14. DELAY .2 Seconds 15. Set HITL Detector Input 11 = On	9.	WHILE phaseStatu	usGroupPhaseOns.1 ≠ 0x14 AND	
10.DELAY 1 Second11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On		phaseStatusGroup	PhaseOns.2 ≠ 0X00	
11.GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.212.WEND Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On	10.	DELAY 1 Seco	nd	
phaseStatusGroupPhaseOns.2 12. WEND Note: Wait until controller reaches 3+5. 13. Set HITL Detector Input 3 = Off 14. DELAY .2 Seconds 15. Set HITL Detector Input 11 = On	11.	GET phaseStat	usGroupPhaseOns.1,	
12.WEND Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On		phaseStatusGr	oupPhaseOns.2	
Note: Wait until controller reaches 3+5.13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On	12.	WEND		
13.Set HITL Detector Input 3 = Off14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On		Note: Wait until co	ntroller reaches 3+5.	
14.DELAY .2 Seconds15.Set HITL Detector Input 11 = On	13.	Set HITL Detector	Input 3 = Off	
15. Set HITL Detector Input 11 = On	14.	DELAY .2 Seconds	5	
	15.	Set HITL Detector	Input 11 = On	

16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x14 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 ≠ 0x02	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls 1	
21	WEND	
22	VERIEY phaseStatusGroupGreens 1 = 0x14 AND	Pass/Fail
 .	phaseStatusGroupGreens 2 = 0x00 AND	1 400/1 41
	phaseStatusGroupVehCalls 1 AND $0x02 = 0x02$	
	Note: Ensure that Phase 2 registers a Vehicle Call during 3+5 Green	
23	Set HITL Detector Input 11 = Off	
20.	DELAY 2 Seconds	
Detector 11 calls P	base 2 when Phase 2 not Green (In 4+5 Green)	
	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 \pm 0x03 AND ringStatus 2 \pm 0x03	
2.	DELAV 1 Second	
J.	DELAT I Secoliu	
4. 5		
5.	WEND	
	Nata Laan until aantrallar roota aanau hara	
C		
0.	Set HILL Detector Input 4 = On	
1.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 11 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x18 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 ≠ 0x02	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 = 0x02	
	<i>Note:</i> Ensure that Phase 2 registers a Vehicle Call during 4+5 Green.	
23.	Set HITL Detector Input 11 = Off	

24.	DELAY .2 Seconds	
Detector 11 calls P	hase 2 when Phase 2 not Green (In 6+12 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 6+12.	
22.	Set HITL Detector Input 6 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 11 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x08 AND	
	phaseStatusGroupVenCalls.1 AND 0x02 ≠ 0x02	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
20	phaseStatusGroupVenCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fall
	phaseStatusGroupGreens.z = 0x00 AND phaseStatusGroupVobCalle 1 AND 0x02 = 0x02	
	phaseolalusoloupvelloalis. 1 AND 0x02 - 0x02	
	Note: Ensure that Phase 2 registers a Vehicle Call during 6+12	
	Green	
32	Set HITL Detector Input 11 = Off	
<u>3</u> 2	DELAY 2 Seconds	
	BEEKT / E OOOONGO	

Detector 11 calls P	Phase 2 when Phase 2 not Green (In 6+11 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
•		
	Note: Loop until controller rests in green somewhere	
6.	Set HITL Detector Input 3 = On	
7	DELAY 2 Seconds	
8	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
0. Q	WHILE phaseStatusGroupPhaseOns $1 \neq 0x14$ AND	
0.	nhaseStatusGroupPhaseOns $2 \neq 0x00$	
10	DELAY 1 Second	
10.	GET nhaseStatusGrounPhaseOns 1	
11.	nhaseStatusGrounPhaseOns 2	
12	WEND	
12.	Note Wait until controller reaches 3+5	
13	Set HITL Detector Input $3 = Off$	
11	DELAV 2 Seconds	
14.	Set HITL Detector Input 6 - On	
10.	DELAV 2 Secondo	
10.	DELAT .2 Seconds	
17.	WHILE phaseStatusGloupFildSeOfis.1, pildSeStatusGloupFildSeOfis.2	
10.	while phaseStatusGroupPhaseOns. $1 \neq 0x20$ And phaseStatusCroupPhaseOns. $2 \neq 0x04$	
10	DELAX 4 Second	
19.	DELAY I Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
۷۱.	WEND Nata Wait until controller reaches 6 / 11	
<u> </u>	Set HIL Detector input 6 = OII	
23.	DELAY .2 Seconds	
<u> </u>	Set HITL Detector input IT = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
07		
27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x04 AND phaseStatusGroupVehCalle 1 AND 0x02 ± 0x02	
	DELAX 4 Constants. TAND 0x02 ≠ 0x02	
28	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
20		
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fall
	phaseStatusGroupUchCalle 1 AND 0x02 = 0x02	
	priasestatus Group vencalis. TAND 0x02 – 0x02	
	Noto: Ensure that Phase 2 registers a Vehicle Call during 6+11	
	Green	
32	Set HITL Detector Input 11 = Off	
 २२	DELAY 2 Seconds	
Detector 11 calls P	Decar 2 when Phase 2 not Green (In 5±0 Green)	
	GET ringStatus 1 ringStatus 2	
۱. ۲	WHILE ringStatus 1 \neq 0v03 AND ringStatus 2 \neq 0v03	
۷.	$VIIILE IIIIgotatus. I \neq 0x00 AND IIIIgotatus. Z \neq 0x00$	

3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 7 = On	
16	DELAY 2 Seconds	
17	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
18	WHILE phaseStatusGroupPhaseOns $1 \neq 0x10$ AND	
10.	$haseStatusGroupPhaseOns 2 \neq 0x01$	
10	DELAY 1 Second	
20	GET nhaseStatusGrounPhaseOns 1	
20.	nhaseStatusGrounPhaseOns 2	
21		
21.	Note: Wait until controller reaches 5+9	
22	Set HITL Detector Input 7 = Off	
23	DELAY 2 Seconds	
20.	Set HITL Detector Input 11 = On	
25	DELAY 2 Seconds	
20.	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
20.	nhaseStatusGrounVehCalls 1	
27	WHILE phaseStatusGroupGreens 1 = 0x10 AND	
21.	phaseStatusGroupGreens 2 = 0x01 AND	
	phaseStatusGroupVehCalls 1 AND $0x02 \neq 0x02$	
28	DELAY 1 Second	
29	GET phaseStatusGroupGreens 1_phaseStatusGroupGreens 2	
20.	nhaseStatusGroupVehCalls 1	
30	WEND	
31	VERIEY phaseStatusGroupGreens 1 = 0x10 AND	Pass/Fail
01.	phaseStatusGroupGreens 2 = 0x01 AND	
	phaseStatusGroupVehCalls 1 AND $0x02 = 0x02$	
	<i>Note:</i> Ensure that Phase 2 registers a Vehicle Call during 5+9 Green.	
32.	Set HITL Detector Input 11 = Off	
33.	DELAY 2 Seconds	
Detector 11 calls P	Phase 2 when Phase 2 not Green (In 1+6 Green)	
1.	GET ringStatus.1. ringStatus.2	
2	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
3	DELAY 1 Second	
4	GET ringStatus 1, ringStatus 2	
5	WEND	
<u>.</u>		

	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns 1 ≠ 0x21 AND	
	phaseStatusGroupPhaseOns 2 ≠ 0X00	
10	DFLAY 1 Second	
11	GET phaseStatusGroupPhaseOns 1	
	phaseStatusGroupPhaseOns 2	
12	WEND	
	Note: Wait until controller reaches 1+6.	
13	Set HITL Detector Input 6 = Off	
14	DELAY 2 Seconds	
15	Set HITL Detector Input 11 = On	
16	DELAY 2 Seconds	
17	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
17.	nhaseStatusGrounVehCalls 1	
18	WHILE phaseStatusGroupGreens $1 = 0x21$ AND	
10.	phaseStatusGroupGreens $2 = 0x00$ AND	
	phaseStatusGroupVehCalls 1 AND $0x02 \neq 0x02$	
19	DELAY 1 Second	
20	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
20.	phaseStatusGroupVehCalls 1	
21	WEND	
22	VERIEY phaseStatusGroupGreens 1 = 0x21 AND	Pass/Fail
 .	$rac{1}{2}$ haseStatusGroupGreens 2 = 0x00 AND	1 400/1 41
	phaseStatusGroupVehCalls 1 AND $0x02 = 0x02$	
	<i>Note:</i> Ensure that Phase 2 registers a Vehicle Call during 1+6 Green.	
23.	<i>Note:</i> Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off	
23. 24.	<i>Note:</i> Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds	
23. 24. Detector 11 calls F	<i>Note:</i> Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds Phase 2 when Phase 2 not Green (In 1+7 Green)	
23. 24. Detector 11 calls F	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds Phase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus 1, ringStatus 2	
23. 24. Detector 11 calls F 1. 2.	<i>Note:</i> Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds Phase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
23. 24. Detector 11 calls F 1. 2. 3.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	
23. 24. Detector 11 calls F 1. 2. 3. 4	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds 'hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus 1 ringStatus 2	
23. 24. Detector 11 calls F 1. 2. 3. 4.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green.Set HITL Detector Input 11 = OffDELAY .2 SecondsPhase 2 when Phase 2 not Green (In 1+7 Green)GET ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03DELAY 1 SecondGET ringStatus.1, ringStatus.2WEND	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere.	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds 'hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds 'hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds 'hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 5. 6. 7. 8. 9	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE nbaseStatusGroupPhaseOns 1 ≠ 0x41 AND	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds Phase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green.Set HITL Detector Input 11 = OffDELAY .2 SecondsPhase 2 when Phase 2 not Green (In 1+7 Green)GET ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03DELAY 1 SecondGET ringStatus.1, ringStatus.2WENDNote: Loop until controller rests somewhere.Set HITL Detector Input 7 = OnDELAY .2 SecondsGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 ANDphaseStatusGroupPhaseOns.2 ≠ 0X00DELAY 1 Second	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds 'hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.1 ≠ 0x40 DELAY 1 Second GET phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green.Set HITL Detector Input 11 = OffDELAY .2 Seconds 'hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03DELAY 1 SecondGET ringStatus.1, ringStatus.2WENDNote: Loop until controller rests somewhere.Set HITL Detector Input 7 = OnDELAY .2 SecondsGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 ANDphaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, phaseSta	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green.Set HITL Detector Input 11 = OffDELAY .2 SecondsPhase 2 when Phase 2 not Green (In 1+7 Green)GET ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03DELAY 1 SecondGET ringStatus.1, ringStatus.2WENDNote: Loop until controller rests somewhere.Set HITL Detector Input 7 = OnDELAY .2 SecondsGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 ANDphaseStatusGroupPhaseOns.2 ≠ 0X00DELAY 1 SecondGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WEND	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 11.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green.Set HITL Detector Input 11 = OffDELAY .2 Secondshase 2 when Phase 2 not Green (In 1+7 Green)GET ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03DELAY 1 SecondGET ringStatus.1, ringStatus.2WENDNote: Loop until controller rests somewhere.Set HITL Detector Input 7 = OnDELAY .2 SecondsGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 ANDphaseStatusGroupPhaseOns.2 ≠ 0X00DELAY 1 SecondGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WENDNote: Wait until controller reaches 1+7.	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12. 13.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 11. 12. 13. 14	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 11. 12. 13. 14. 15.	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds Phase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 \neq 0x41 AND phaseStatusGroupPhaseOns.2 \neq 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 7 = Off DELAY .2 Seconds	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 11. 12. 13. 14. 15. 16	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds 'hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 11 = On DELAY .2 Seconds	
23. 24. Detector 11 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 11. 12. 13. 14. 15. 16. 17	Note: Ensure that Phase 2 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 11 = Off DELAY .2 Seconds 'hase 2 when Phase 2 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 7 = Off DELAY .2 Seconds GET phaseStatusGroupGreens 1 phaseStatusGroupGreens 2	

	nhaseStatusGrounVehCalls 1	
18	WHILE phaseStatusGroupGreens $1 = 0x41 \text{ AND}$	
10.	$r_{\text{A}} = 0.41 \text{ AND}$	
	phaseStatusGroupVebCalle 1 AND $0x02 \pm 0x02$	
10	DELAV 1 Second	
18. 20	CET phone Status Croup Croope 1, phone Status Croup Croope 2	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
04		
21.		
22.	VERIFY phaseStatusGroupGreens.1 = 0x41 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVenCalls.1 AND 0x02 = 0x02	
	Note: Ensure that Phase 2 registers a Vehicle Call during 1+7 Green.	
23.	Set HITL Detector Input 11 = Off	
24.	DELAY .2 Seconds	
Detector 11 calls P	hase 2 when Phase 2 not Green (In 1+8 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6	Set HITL Detector Input 8 = On	
7	DELAY 2 Seconds	
<u> </u>	GET nhaseStatusCrounPhaseOns 1 nhaseStatusCrounPhaseOns 2	
0.	$WHII E phaseStatusGroupPhaseOns 1 \neq 0x81 AND$	
9.	while phase status Group Phase Ons $2 \neq 0.000$	
10	DELAV 1 Second	
10.	DELAT I Secoliu	
11.	GET phaseStatusGroupPhaseOns.1,	
40	phaseStatusGroupPhaseOns.2	
12.	WEND Na (a. Wait watil as a traller researce at 10	
	Note: Walt until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 11 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x81 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 ≠ 0x02	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x81 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x02 = 0x02	
	· · ·	
	Note: Ensure that Phase 2 registers a Vehicle Call during 1+8 Green.	
23.	Set HITL Detector Input 11 = Off	
24.	DELAY .2 Seconds	

Detector 11 calls Phase 2 when Phase 2 not Green (In 1+13 Green)		
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
•		
	Note: Loop until controller rests in green somewhere.	
6	Set HITL Detector Input 8 = On	
7	DELAY 2 Seconds	
8	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
<u>9</u>	WHILE phaseStatusGroupPhaseOns $1 \neq 0.081$ AND	
0.	phaseStatusGroupPhaseOns $2 \neq 0x00$	
10	DELAY 1 Second	
11	GET nhaseStatusGrounPhaseOns 1	
	nhaseStatusGrounPhaseOns 2	
12	WEND	
12.	Note: Wait until controller reaches 1+8	
13	Set HITL Detector Input 8 = Off	
11	DELAX 2 Seconds	
15	Set HITL Detector Input 3 - On	
10.	DELAX 2 Seconds	
10.	DELAT .2 Seconds	
17.	GET phaseStatusGroupPhaseOns. 1, phaseStatusGroupPhaseOns.2	
10.	while phase status Group Phase Ons $2 \neq 0.001$ And	
10	DELAX 4 Second	
19.	DELAT I Secollu	
20.	GET phaseStatusGroupPhaseOns.1,	
۷۱.	Victo: Wait until controller reaches 1+13	
22	Set HITL Detector Input 3 - Off	
22.	DELAV 2 Seconde	
23.	Set HITL Detector Input 11 - On	
24.	DELAX 2 Secondo	
20	DELAT .2 Seconds	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	WILLIE nhoosStatusCroupCroops 1 = 0x01 AND	
27.	while phasestatusGroupGreens. I = 0x01 AND	
	phaseStatusGroupVabCalls 1 AND $0x02 \neq 0x02$	
<u></u>	DELAV 1 Second	
20.	CET phaseStatusCroupCroops 1, phaseStatusCroupCroops 2	
29.	BET phaseStatusGroupSteens.1, phaseStatusGroupGreens.2,	
20		
JU. 21	VEND	
JI.	vERIFY phaseStatusGroupGreens.1 = 0x01 AND	Pass/Fall
	phaseStatusGroupVabCalls 1 AND 0x02 - 0x02	
	priasestatus sroup vencalis. TAND 0x02 - 0x02	
	Note: Ensure that Phase 2 registers a Vehicle Call during 1+12	
	Green	
30	Set HITL Detector Input 11 = Off	
32	DELAV 2 Seconde	
55.		

Detector 11 extends Phase 2 when Phase 2 is Green until a gap in Detector 11 activity occurs at				
which time Detector 11 becomes inactive until Phase 2 Yellow				
1.	GET vehicleDetectorExtend.11 and RECORD RESPONSE VALUE			
	in [currentExtendValue]			
2.	SET vehicleDetectorExtend.11 = 40			
	<i>Note:</i> Set Detector 11 extend time = 4 seconds.			
3.	GET ringStatus.1, ringStatus.2			
4.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03			
5.	DELAY 1 Second			
6.	GET ringStatus.1, ringStatus.2			
7.	WEND			
	Note: Loop until controller rests somewhere.			
8.	Set HITL Detector Input 6 = On			
9.	DELAY .2 Seconds			
10.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2			
11.	WHILE phaseStatusGroupPhaseOns $1 \neq 0x21$ AND			
	phaseStatusGroupPhaseOns.2 ≠ 0x00			
12	DELAY 1 Second			
13	GET nhaseStatusGrounPhaseOns 1			
10.	nhaseStatusGrounPhaseOns 2			
14	WEND			
17.	Note: Wait until controller reaches 1+6			
15	Set HITL Detector Input 6 - Off			
16	DELAX 2 Seconde			
10.	DELAT .2 Seconds			
17.	Set HTL Detector input TT = On (To call and then extend Phase 2) $DELAY = 0$			
18.	DELAY .2 Seconds			
19.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2			
20.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND			
	phaseStatusGroupPhaseOns.2 ≠ 0x00			
21.	DELAY 1 Second			
22.	GET phaseStatusGroupPhaseOns.1,			
	phaseStatusGroupPhaseOns.2			
23.	WEND			
	Note: Wait until controller reaches 2+5.			
24.	Set HITL Detector Input 7 = On (Detector 11 will extend Phase 2			
	when there is an opposing call)			
25.	DELAY .2 Seconds			
26.	GET ringStatus.1			
27.	WHILE ringStatus.1 AND 0x07 ≠ 0x01 (xxxxx001 = extension)			
28.	DELAY .1 Second			
29.	GET ringStatus.1			
30.	WEND			
	<i>Note:</i> Wait for indication that extensions are timing.			
	Ring 1 = 2 , 3, 4, 9, 11, 12, & 1			
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13			
31.	GET phaseStatusGroupPhaseOns.1			
32.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail		
	AND 0x02 = 0x02			
	Note: Verify that the extensions are on Phase 2.			
33.	Set HITL Detector Input 11 = Off			
34.	DELAY 3 Seconds			
-----	--	-----------		
35.	Set HITL Detector Input 11 = On			
36.	DELAY .2 Seconds			
37.	Set HITL Detector Input 11 = Off			
38.	DELAY 2.8 Seconds			
39.	GET ringStatus.1			
40.	VERIFY that RESPONSE VALUE ringStatus.1 AND 0x07 = 0x01			
	(xxxxx001 = extension)			
	<i>Note:</i> Ring 1 = 2 , 3, 4, 9, 11, 12, & 1			
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13			
41.	GET phaseStatusGroupPhaseOns.1			
42.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail		
	AND 0x02 = 0x02			
	<i>Note:</i> Verify that extensions are still timing on Phase 2 because call			
	was entered < 4 seconds later.			
43.	Set HITL Detector Input 11 = On			
44.	DELAY .2 Seconds			
45.	Set HITL Detector Input 11 = Off			
46.	DELAY 4.8 Seconds			
	<i>Note:</i> Since the time between actuations is now 5 seconds, the timer			
	will gap and therefore disable Detector 11 from putting in any further			
	extensions.			
47.	Set HITL Detector Input 11 = On			
48.	DELAY .2 Seconds			
49.	GET ringStatus.1			
50.	VERIFY that RESPONSE VALUE ringStatus.1 AND 0x07 = 0x04			
	(xxxxx100 = Yellow Change)			
	<i>Note:</i> Verify that the Yellow Change is on Phase 2 because call was			
	entered > 4 seconds later.			
	Ring 1 = 2 , 3, 4, 9, 11, 12, & 1			
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13			
51.	GET phaseStatusGroupPhaseOns.1, ringStatus.1,			
	phaseStatusGroupVehCalls.1			
52.	WHILE (phaseStatusGroupPhaseOns.1 AND 0x02 = 0x02) AND			
	(ringStatus.1 AND 0x07 = 0x04) AND (phaseStatusGroupVehCalls.1			
	AND 0x02 ≠ 0x02)			
53.	DELAY 1 Second			
54.	GET phaseStatusGroupPhaseOns.1, ringStatus.1,			
	phaseStatusGroupVehCalls.1			
55.	WEND			
56.	VERIFY (phaseStatusGroupPhaseOns.1 AND 0x02 = 0x02) AND	Pass/Fail		
	(ringStatus.1 AND 0x07 = 0x04) AND (phaseStatusGroupVehCalls.1			
	AND $0x02 = 0x02$)			
	Note: Ensure that Phase 2 registers a Vehicle Call during 2 Yellow.			
57.	Set HITL Detector Input 7 = Off and 11 = Off			
58.	DELAY .2 Seconds			
59.	SET vehicleDetectorExtend.11 = [currentExtendValue],			
	Note: Restore original values.			

Teardown				
	PERFORM Detector Teardown – TC020 if not proceeding to another detector operation test case			
	Test Case	Results		
Tested By:		Date Tested		Pass/Fail
Test Case Notes:	<notes> Test Case 12 does not p</notes>	ass an eq	uivalent	
Version History:	v1.00 05/09/06 Initial Draft – RD v1.01 07/05/06 Updated notes – v1.02 07/12/06 Implemented scr	R RDR ipt and pr	oofed – JJ	

Detector 12 Operations

Tost Casa:	Titlo:	Detector 12 Operations	
	Description:	Verifies the operation of Detector 12 to call and ext	and Phase 4
10012	Description.	under appoint on Delector 12 to call and extend interval 4516	(6 ± 12)
	Constanta	under specific conditions and exterior interval 4510	(0+12).
	Constants.	a sum and the state of the last	
	variables:		
	Pass/Fall	The DUT shall pass every verification step included	a within the
		Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detect	or Operations Setup – TC019 if not already done	
	SO.		
Detector 12 calls P	hase 4 when Phase	e 4 not Green (In 6+12 Green)	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus.	$1 \neq 0x03$ AND ringStatus. $2 \neq 0x03$	
3.	DELAY 1 Seco	nd	
4.	GET ringStatus	.1, ringStatus.2	
5.	WEND		
	Note: Loop until co	ntroller rests somewhere.	
6.	Set HITL Detector	Input 4 = On	
7.	DELAY .2 Seconds	5	
8.	GET phaseStatus	GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatu	usGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroup	PhaseOns.2 ≠ 0X00	
10.	DELAY 1 Seco	nd	
11.	GET phaseStat	usGroupPhaseOns.1.	
	phaseStatusGr	oupPhaseOns.2	
12.	WEND		
	Note: Wait until co	ntroller reaches 4+5.	
13.	Set HITL Detector	Input 4 = Off	
14.	DELAY .2 Seconds	5	
15.	GET ringStatus 1	ringStatus.2	
16.	WHILE ringStatus	$1 \neq 0x03$ AND ringStatus.2 $\neq 0x03$	
10. 11. 12. 13. 14. 15. 16.	DELAY 1 Secon GET phaseStat phaseStatusGro WEND <i>Note:</i> Wait until con Set HITL Detector DELAY .2 Seconds GET ringStatus.1, WHILE ringStatus.	nd usGroupPhaseOns.1, oupPhaseOns.2 ntroller reaches 4+5. Input 4 = Off s ringStatus.2 $1 \neq 0x03$ AND ringStatus.2 $\neq 0x03$	

17.	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
-		
	<i>Note:</i> Wait for 4+5 Green Rest.	
20.	Set HITL Detector Input 6 = On	
21.	DELAY .2 Seconds	
22	GET phaseStatusGroupPhaseOns 1 phaseStatusGroupPhaseOns 2	
23	WHILE phaseStatusGroupPhaseOns $1 \neq 0x20$ AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x08	
24	DELAY 1 Second	
25	GET phaseStatusGroupPhaseOns 1	
_0.	phaseStatusGroupPhaseOns.2	
26	WEND	
	Note: Wait until controller reaches 6+12	
27	Set HITL Detector Input 6 = Off	
28	DELAY 2 Seconds	
29	Set HITL Detector Input 12 = On	
30	DELAY 2 Seconds	
31	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
01.	nhaseStatusGrounVehCalls 1	
32	WHILE nhaseStatusGrounGreens $1 = 0x20 \text{ AND}$	
52.	nhaseStatusGrounGreens 2 = 0x08 AND	
	phaseStatusGroupVehCalls 1 AND $0x08 \neq 0x08$	
33	DELAY 1 Second	
30. 31	GET phaseStatusGroupGreens 1 phaseStatusGroupGreens 2	
54.	obaseStatusGroupVebCalls 1	
25	WEND	
36	VEDIEV phaseStatusGroupGreeps 1 - 0x20 AND	Dass/Eail
50.	$v = 1 \times 10^{-10} \text{ mascolatusOroupOreens}$ $1 = 0 \times 20 \text{ AND}$	r ass/r all
	phaseStatusGroupVebCalls 1 AND 0x08 = 0x08	
	<i>Note:</i> Ensure that Phase 4 registers a Vehicle Call during 6+12	
	Green.	
37.	Set HITL Detector Input 12 = Off	
38.	DELAY .2 Seconds	
Detector 12 calls F	Phase 4 when Phase 4 not Green (In 6+11 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	

12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
	DELAY .2 Seconds	
	GET ringStatus.1, ringStatus.2	
	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
17.	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Wait for 3+5 Green Rest.	
20.	Set HIL Detector Input 6 = On	
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x04	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
~	Note: Wait until controller reaches 6+11.	
27.	Set HIIL Detector Input 6 = Off	
28.	DELAY .2 Seconds	
29.	Set HIIL Detector Input 12 = On	
30.	DELAY .2 Seconds	
31.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
32.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVenCalls.1 AND 0x08 ≠ 0x08	
33.	DELAY 1 Second	
34.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	pnaseStatusGroupVenCalls.1	
35.		D/E-1
36.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fall
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVenCalls.1 AND 0x08 = 0x08	
	Noto: Ensure that Dhase 4 registers a Vahiala Call during 6+11	
	Creen	
27	Set HITL Detector Input 12 - Off	
	DELAV 2 Seconds	
JO. Detector 12 calls F	DELAT 2 Seconds	
	GET ringStatus 1 ringStatus 2	
່. ົ	WHILE ringStatus 1 \pm 0x03 AND ringStatus 2 \pm 0x03	
<u>ک.</u> ۲	DEL AV 1 Second	
<u>J</u>	CET ringStatus 1 ringStatus 2	
<u>+.</u> Б		
J.		
	Note: Loop until controller rests in green somewhere	
6	Set HITI Detector Input $3 = \Omega n$	
7	DELAV 2 Seconde	
۲. و	CET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
U. U.		

9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
17.	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Wait for 3+5 Green Rest.	
20	Set HITL Detector Input 7 = On	
21	DELAY 2 Seconds	
22	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
22.	WHILE phaseStatusGroupPhaseOns 1 \pm 0x10 AND	
25.	while phase Status Group Phase One $2 \neq 0x01$	
24	DELAV 1 Second	
24.	DELAT I SECUIU	
20.	GET phaseStatusGroupPhaseOns.1,	
06	phaseStatusGroupPhaseOns.2	
20.	VVEND Nata Wait until controller reaches ELO	
07		
27.	Set HILL Detector Input 7 = Off	
28.	DELAY .2 Seconds	
29.	Set HITL Detector Input 12 = On	
30.	DELAY .2 Seconds	
31.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
32.	WHILE phaseStatusGroupGreens.1 = 0x10 AND	
	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
33.	DELAY 1 Second	
34.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
35.	WEND	
36.	VERIFY phaseStatusGroupGreens.1 = 0x10 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
	Note: Ensure that Phase 4 registers a Vehicle Call during 5+9 Green.	
37.	Set HITL Detector Input 12 = Off	
38.	DELAY .2 Seconds	
Detector 12 calls P	hase 4 when Phase 4 not Green (In 1+6 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4	GET ringStatus.1, ringStatus.2	
5	WEND	
Ŭ.		
	Note: Loop until controller rests somewhere.	

6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns 1	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13	Set HITL Detector Input 6 = Off	
14	DELAY 2 Seconds	
15	Set HITL Detector Input 12 = On	
10.	DELAY 2 Seconds	
10.	CET phaseStatusCroupCreeps 1 phaseStatusCroupCreeps 2	
17.	ber phaseStatusGroupVebCalle 1	
10	WHILE phaseStatusGroupGroops 1 = 0v21 AND	
10.	nhasoStatusGroupGroops 2 = 0x00 AND	
	phaseStatusGroupVabCalle 1 AND $0x08 \neq 0x08$	
10	DELAV 1 Second	
ାଞ. ୦୦	DELAT I Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
21.		
22.	VERIFY phaseStatusGroupGreens.1 = 0x21 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
	Matter Example that Dharas Annalistana a Mahiala Oall during 410 One su	
	<i>Note:</i> Ensure that Phase 4 registers a Vehicle Call during 1+6 Green.	
23.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off	
23. 24.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds	
23. 24. Detector 12 calls F	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds Phase 4 when Phase 4 not Green (In 1+7 Green)	
23. 24. Detector 12 calls F	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds Phase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2	
23. 24. Detector 12 calls F 1. 2.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green.Set HITL Detector Input 12 = OffDELAY .2 SecondsPhase 4 when Phase 4 not Green (In 1+7 Green)GET ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
23. 24. Detector 12 calls F 1. 2. 3.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds Phase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	
23. 24. Detector 12 calls F 1. 2. 3. 4.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds 'hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds Phase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds Phase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds 'hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere.	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds 'hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds 'hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds 'hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 5. 6. 7. 8. 9.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds 'hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 5. 6. 7. 8. 9.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green.Set HITL Detector Input 12 = OffDELAY .2 SecondsPhase 4 when Phase 4 not Green (In 1+7 Green)GET ringStatus.1, ringStatus.2WHILE ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03DELAY 1 SecondGET ringStatus.1, ringStatus.2WENDNote: Loop until controller rests somewhere.Set HITL Detector Input 7 = OnDELAY .2 SecondsGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 ANDphaseStatusGroupPhaseOns.2 ≠ 0X00	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green.Set HITL Detector Input 12 = OffDELAY .2 SecondsPhase 4 when Phase 4 not Green (In 1+7 Green)GET ringStatus.1, ringStatus.2WHILE ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03DELAY 1 SecondGET ringStatus.1, ringStatus.2WENDNote: Loop until controller rests somewhere.Set HITL Detector Input 7 = OnDELAY .2 SecondsGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 ANDphaseStatusGroupPhaseOns.2 ≠ 0X00DELAY 1 Second	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds 'hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds *hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, ≠ 0X00	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, phaseStatusGroupPhaseOns.2, wence WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, wence WEND Note: Wait until controller reaches 1+7.	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 11. 12. 13.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds 'hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 11. 12. 13. 14.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 11. 12. 13. 14. 15.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 \neq 0x41 AND phaseStatusGroupPhaseOns.2 \neq 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, WEND <i>Note:</i> Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 7 = Off DELAY .2 Seconds	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 11. 12. 13. 14. 15. 16	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds 'hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHLE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 12 = On DELAY .2 Seconds Set HITL Detector Input 12 = On DELAY .2 Seconds	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 11. 12. 13. 14. 15. 16. 17	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHLE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHD Note: Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 12 = On DELAY .2 Seconds GET nhaseStatusGroupGreens 1 nhaseStatusGroupGreens 2	
23. 24. Detector 12 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17.	Note: Ensure that Phase 4 registers a Vehicle Call during 1+6 Green. Set HITL Detector Input 12 = Off DELAY .2 Seconds 'hase 4 when Phase 4 not Green (In 1+7 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+7. Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 7 = Off DELAY .2 Seconds Set HITL Detector Input 12 = On DELAY .2 Seconds GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls 1 <td></td>	

18.	WHILE phaseStatusGroupGreens.1 = 0x41 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVebCalls 1 AND 0x08 ≠ 0x08	
19	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
21	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x41 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	Pass/Fail
	Note: Ensure that Phase 4 registers a Vehicle Call during 1+7 Green	
23	Set HITI Detector Input 12 = Off	
20.	DELAY 2 Seconds	
Detector 12 calls P	Phase 4 when Phase 4 not Green (In 1+8 Green)	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1. ringStatus.2	
5.	WEND	
_		
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
12.	WEND <i>Note:</i> Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 12 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x81 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x81 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x08 = 0x08 <i>Note:</i> Ensure that Phase 4 registers a Vehicle Call during 1+8 Green.	Pass/Fail
23.		
24.	Set HITL Detector Input 12 = Off	
25.	DELAY .2 Seconds	
Detector 12 calls P	hase 4 when Phase 4 not Green (In 2+16 Green)	

1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Wait for 1+8 Green Rest.	
20.	Set HITL Detector Input 2 = On	
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0x80	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
	No (c.)Mait controller constants of a O (40)	
07	Note: Walt until controller reaches 2+16.	
27.	Set HILL Detector input $2 = Off$	
<u> </u>	DELAT .2 Seconds	
<u> </u>	Set HITL Detector input $12 = On$	
30.	DELAY .2 Seconds	
31.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
20		
32.	while phasestatusGroupGreens. 1 - 0x02 AND	
	phaseStatusGroupVebCalls 1 AND $0x08 \neq 0x08$	
33	DELAY 1 Second	
30. 31	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
J 4 .	nhaseStatusGrounVehCalle 1	
32	WEND	
JJJ.		

36.	VERIEY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
00.	nhaseStatusGrounGreens 2 = 0x80 AND	
	phaseStatusGroupVebCalls 1 Δ ND 0x08 = 0x08	
	phaseotatusoroupverioalis. I AND 0x00 - 0x00	
	Note: Ensure that Phase 4 registers a Vehicle Call during 2+16	
	Green	
37	Set HITL Detector Input 12 = Off	
	DELAV 2 Seconds	
Detector 12 calls B	DELAT 2 0000103	
	CET ringStatue 1 ringStatue 2	[
۱. ۲	GET IIIIgoldius. I, IIIigoldius. Z WEIII E ringStatus 1, \pm 0x02 AND ringStatus 2, \pm 0x02	
<u> </u>	DELAX 4 Second	
<u>ک</u> .	DELAT I Secoliu	
4. F		
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
<u> </u>	Set HILL Detector Input 7 = On	
1.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	GET ringStatus.1, ringStatus.2	
16.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
17.	DELAY 1 Second	
18.	GET ringStatus.1, ringStatus.2	
19.	WEND	
	Note: Wait for 1+7 Green Rest.	
20.	Set HITL Detector Input 2 = On	
21.	DELAY .2 Seconds	
22.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
23.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x40	
24.	DELAY 1 Second	
25.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
26.	WEND	
	Note: Wait until controller reaches 2+15.	
27.	Set HITL Detector Input 2 = Off	
28.	DELAY .2 Seconds	
29.	Set HITL Detector Input 12 = On	
30.	DELAY .2 Seconds	
31.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2.	
	phaseStatusGroupVehCalls.1	
32.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
d	· · · · · · · · · · · · · · · · · · ·	

33.	DELAY 1 Second	
34.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
35.	WEND	
36.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
	Note: Ensure that Phase 4 registers a Vehicle Call during 2+15	
	Green.	
37.	Set HITL Detector Input 12 = Off	
38.	DELAY 2 Seconds	
Detector 12 calls P	Phase 4 when Phase 4 not Green (In 1+13 Green)	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
<u>ک</u> .	DELAY 1 Second	
3. 4	GET ringStatus 1 ringStatus 2	
5	WEND	
5.	WEND	
	Note: Loop until controller rests in green somewhere	
6	Set HITL Detector Input 8 = On	
7	DELAY 2 Seconds	
2. 2	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
0. 0	WHILE phaseStatusGroupPhaseOns $1 \neq 0.081$ AND	
5.	nhaseStatusGroupPhaseOns $2 \neq 0x00$	
10	DELAY 1 Second	
10.	GET nhaseStatusGrounPhaseOns 1	
11.	nhaseStatusGrounPhaseOns 2	
12	WEND	
12.	Note: Wait until controller reaches 1+8	
13	Set HITL Detector Input 8 = Off	
10.	DELAY 2 Seconds	
15	GET ringStatus 1 ringStatus 2	
10.	WHILE ringStatus 1 \pm 0x03 AND ringStatus 2 \pm 0x03	
10.	DELAV 1 Second	
17.	GET ringStatus 1 ringStatus 2	
10.		
19.	WEND	
	Note: Wait for 1+8 Green Rest	
2∩	Set HITL Detector Input $3 = On$	
20.	DELAY 2 Seconds	
21.	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
22.	WHILE nhaseStatusGroupPhaseOns 1 \pm 0v01 AND	
23.	while phaseStatusGroupPhaseOns $2 \neq 0x10$	
	DELAY 1 Second	
25	GET nhaseStatusGrounPhaseOns 1	
20.	nhaseStatusGrounPhaseOns 2	
26	WEND	
20.	Note: Wait until controller reaches 1+13	
27	Set HITL Detector Input $3 = Off$	
21.	DELAY 2 Seconds	
20.	Sat HITL Detector Input $12 = \Omega n$	
29. 20	DELAV 2 Seconde	
ა	DELAT 2 SECONDS	

31.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
32.	WHILE phaseStatusGroupGreens.1 = 0x01 AND	
	phaseStatusGroupGreens.2 = 0x10 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
33.	DELAY 1 Second	
34.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
35.	WEND	
36.	VERIFY phaseStatusGroupGreens.1 = 0x01 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x10 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
	Note: Ensure that Phase 4 registers a Vehicle Call during 1+13	
	Green.	
37.	Set HITL Detector Input 12 = Off	
38.	DELAY .2 Seconds	
Detector 12 calls P	hase 4 when Phase 4 not Green (In 2+5 Green)	
1.	GET ringStatus.1. ringStatus.2	
2	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
3	DELAY 1 Second	
<u> </u>	GET ringStatus 1 ringStatus 2	
5	WEND	
5.	WEND	
	Note: Loop until controller rests somewhere	
6	Set HITL Detector Input $2 = On$	
7	DELAV 2 Seconds	
<u> </u>	CET nhaseStatusCrounPhaseOns 1_nhaseStatusCrounPhaseOns 2	
0.	$WHII E phaseStatusGroupPhaseOns 1 \neq 0x12 AND$	
9.	while phase status Group Phase Ons. $1 \neq 0.00$	
10	DELAV 1 Second	
10.	CET phaseStatueCroupDhaseOne 1	
11.	GET phaseStatusGroupPhaseOns.1,	
10	phaseStatusGroupPhaseOns.2	
12.	WEND Noto Wait until controller reaches 215	
10		
13.	Set HITL Detector input 2 = OII	
14.	DELAY .2 Seconds	
15.	Set HILL Detector Input 12 = On	
16.	DELAT .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
18.	WHILE phaseStatusGroupGreens.1 = 0x12 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	pnaseStatusGroupVenCalls.1 AND 0x08 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	pnaseStatusGroupVenCalls.1	
21.		
22.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND	Pass/Fail
	pnaseStatusGroupGreens.2 = 0x00 AND	
	pnaseStatusGroupVenCalls.1 AND 0x08 = 0x08	
~-	Ivote: Ensure that Phase 4 registers a Vehicle Call during 2+5 Green.	
23.	Set HITL Detector Input 12 = Off	

24.	DELAY .2 Seconds	
Detector 12 calls P	hase 4 when Phase 4 not Green (In 3+5 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 12 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x14 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x14 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x08 = 0x08	
	Note: Ensure that Phase 4 registers a Vehicle Call during 3+5 Green.	
23.	Set HITL Detector Input 12 = Off	
24.	DELAY .2 Seconds	
Detector 12 extend	Is Phase 4 until Phase 6 call exists and Detector 12 gaps and Phase	e 6 call
continues		
1.		
	[currentExtend value]	
Ζ.	SET vehicleDelectorExtend. 12 = 40	
	Note: Set Detector 12 extend time -4 seconds so that actuations	
	less than A seconds apart keep the phase extending	
3	GET ringStatus 1 ringStatus 2	
J.	WHILE ringStatus 1, \pm 0x03 AND ringStatus 2, \pm 0x03	
<u>+.</u> 5	DEL ΔΥ 1 Second	
6	GET ringStatus 1 ringStatus 2	
7		
1.		
	Note: Loop until controller rests somewhere	
L		

8.	Set HITL Detector Input 8 = On	
9.	DELAY .2 Seconds	
10.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
11.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
12.	DELAY 1 Second	
13.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
14.	WEND	
	Note: Wait until controller reaches 1+8.	
15.	Set HITL Detector Input 8 = Off	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 12 = On (To call and then extend Phase 4	
18.	DELAY .2 Seconds	
19.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
20.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
21.	DELAY 1 Second	
22.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
23.	WEND	
	Note: Wait until controller reaches 4+5.	
24.	Set HITL Detector Input 6 = On (Detector 12 will extend Phase 4	
	when there is an opposing call)	
25.	DELAY .2 Seconds	
	Note: This should get Phase 4 to start timing extensions and sets up	
	conditional logic.	
26.	GET ringStatus.1	
27.	WHILE ringStatus.1 AND $0x07 \neq 0x01$ (xxxxx001 = extension)	
28.	DELAY .1 Second	
29.	GET ringStatus.1	
30.	WEND	
	Note: Wait for indication that extensions are timing.	
	Ring $1 = 2, 3, 4, 9, 11, 12, \& 1$	
	Ring $2 = 15, 16, 5, 6, 7, 8, & 13$	
31.	GET phaseStatusGroupPhaseOns.1	
32.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND $0x08 = 0x08$	
	Note: Varify that the outencience are an Dhane 4	
<u></u>	Note. Verify that the extensions are on Phase 4.	
ురి. 	Sel HIL Delector Input 12 - On	
04. 25	DELAT 5 Seconds	
<u>ວວ</u> .	Sel HIL Delector Input 12 - On	
ა0. ეუ	DELAT .2 Seconds Set HITI Detector Input 12 = Off	
১/. ২০	DELAV 2.9 Seconda	
აბ.	DELAT 2.6 Seconds	
<u>.</u>		
40.	VERIFY THAT RESPONSE VALUE RINGSTATUS.1 AND 0x07 = 0x01	Pass/Fail
	(XXXXXUU'I = extension)	
	Note: Ding $1 - 2, 2, 4, 0, 44, 40, 9, 4$	
	Note: King $T = 2, 3, 4, 9, 11, 12, & 1$	
	κ ing 2 = 15, 16, 5, 6, 7, 8, & 13	

41.	GET phaseStatusGroupPhaseOns.1	
42.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x08 = 0x08	
	<i>Note:</i> Verify that extensions are still timing on Phase 4 because call	
	was entered < 4 seconds later.	
43.	Set HITL Detector Input 12 = On	
44	DELAY 2 Seconds	
45	Set HITL Detector Input 12 = Off	
46	DELAY 2.8 Seconds	
μ - 47	GET ringStatus 1	
<u>/</u> 8	VERIEV that RESPONSE VALUE ringStatus 1 AND $0x07 = 0x01$	Pass/Fail
40.	(xxxxx001 = extension)	1 435/1 41
	N_{a} (a) D_{a} (b) D_{a} (c) D_{a	
	Note: Ring $1 = 2, 3, 4, 9, 11, 12, & 1$	
40	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
49.	GET phaseStatusGroupPhaseOns.1	
50.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND $0x08 = 0x08$	
	<i>Note:</i> Verify that extensions are still timing on Phase 4 because call	
	was entered < 4 seconds later.	
51.	Set HITL Detector Input 12 = On	
52.	DELAY .2 Seconds	
53.	Set HITL Detector Input 12 = Off	
54.	DELAY 4.8 Seconds	
	<i>Note:</i> Since the time between actuations is now 5 seconds, the timer	
	will gap and therefore disable Detector 12 from putting in any further	
	extensions.	
55.	Set HITL Detector Input 12 = On	
56.	DELAY .2 Seconds	
57.	GET ringStatus.1	
58.	VERIFY that RESPONSE VALUE ringStatus.1 AND 0x07 ≠ 0x01	Pass/Fail
	(xxxxx001 = extension)	
	<i>Note:</i> Verify that Phase 4 is no longer timing extensions.	
	Ring 1 = 2, 3, <i>4</i> , 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
59.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupVehCalls.1	
60.	WHILE (phaseStatusGroupPhaseOns.1 AND 0x08 = 0x08) AND	
	(phaseStatusGroupVehCalls.1 AND 0x08 ≠ 0x08)	
61.	DELAY 1 Second	
62.	GET phaseStatusGroupPhaseOns.1,	
•=-	phaseStatusGroupVehCalls.1	
63.	WEND	
64.	VERIFY (phaseStatusGroupPhaseOns.1 AND 0x08 = 0x08) AND	Pass/Fail
	(phaseStatusGroupVehCalls.1 AND 0x08 = 0x08)	
	u	
	Note: Ensure that Phase 4 registers a Vehicle Call.	
65.	Set HITL Detector Input 6 = Off and 12 = Off	
66.	DELAY .2 Seconds	
67	POST-CONDITION The Detector 12 vehicleDetectorExtend is still	
5	set to 4 seconds	

Detector 12 extend	Is Phase 4 until Phase 6 call exists and Detector 12 gaps but not if I	Phase 6 call
uisappears	DRE CONDITION This procedure assumes that Detector 12	
	vehicle Detector Extend in still get to 4 seconds	
	CET ringStatus 1, ringStatus 2	
I.	GET IIIgStatus. 1, IIIgStatus.2	
Z.	WHILE ringStatus.1 7 0x03 AND ringStatus.2 7 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
-	Note: Loop until controller rests somewhere.	
6.	Set HILL Detector Input 8 = On	
7.	DELAY .2 Seconds	
	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 12 = On (To call and then extend Phase 4)	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
18	WHILE phaseStatusGroupPhaseOns 1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
19	DFLAY 1 Second	
20	GET phaseStatusGroupPhaseOns 1	
_0.	phaseStatusGroupPhaseOns 2	
21		
	Note: Wait until controller reaches 4+5	
22	Set HITL Detector Input 6 = On	
23	DELAY 2 Seconds	
20.		
	Note: Detector 12 will extend Phase 4 when there is an opposing	
24	GET ringStatus 1	
25	WHILE ringStatus 1 AND $0x07 \neq 0x01$ (xxxxx001 = extension)	
26	DFLAY 1 Second	
20.	GET ringStatus 1	
27.	WEND	
20.	WEND	
	Note: Wait for indication that extensions are timing	
	Ring $1 = 2$ 3 4 9 11 12 & 1	
	Ring $2 = 15$ 16 5 6 7 8 & 13	
20	GET nhaseStatusGrounPhaseOns 1	
	VEDIEV that DESDONSE VALUE phaseStatusCroupDhaseOne 1	Dass/Fail
50.	$\Delta ND D = 0.08$	r ass/rall
	Note: Verify that the extensions are timing on Phase 4	
21	Sat HITL Detector Input 12 - Off	
ວາ. ວາ	DELAV 2 Seconde	
JZ.		

33.	Set HITL Detector Input 12 = On	
34.	DELAY .2 Seconds	
35.	Set HITL Detector Input 12 = Off	
36	DELAY 2.8 Seconds	
37	GET ringStatus 1	
	VERIEV that RESPONSE VALUE ringStatus 1 AND $0x07 = 0x01$	Pass/Fail
00.	(xxxxx001 = extension)	1 400/1 41
	Ring 1 = 2, 3, 4 , 9, 11, 12, & 1	
	Ring $2 = 15, 16, 5, 6, 7, 8, \& 13$	
39.	GET phaseStatusGroupPhaseOns.1	
40.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x08 = 0x08	
	<i>Note:</i> Verify that extensions are still timing is on Phase 4.	
41.	Set HITL Detector Input 12 = On	
42.	DELAY .2 Seconds	
43.	Set HITL Detector Input 12 = Off	
44.	DELAY 2.8 Seconds	
45.	GET ringStatus.1	
46.	VERIFY that RESPONSE VALUE ringStatus.1 AND 0x07 = 0x01	Pass/Fail
	(xxxxx001 = extension)	
	<i>Note:</i> Ring 1 = 2, 3, <i>4</i> , 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
47.	GET phaseStatusGroupPhaseOns.1	
48.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x08 = 0x08	
	No (a Marife that as too is an a fill timing an Dhara A	
40	Note: Verify that extensions are still timing on Phase 4.	
49.	Set HILL Detector input 12 = On	
50.	DELAY .2 Seconds	
51.	Set HILL Detector input $12 = 0\pi$, $4 = 0n$, $6 = 0\pi$ and $7 = 0n$	
	Note: Setting Detector 4 on will keep extending Dhase 4 irrespective	
	of Detector 12, po call on Phase 6 will resot Detector 12 gap	
	function, and the call on Phase 7 will enable extensions to time	
52	DELAY 4.8 Seconds	
52.	DEEAT 4.0 Seconds	
	Note: This would have the effect of allowing the extend timer to gap	
	and therefore disable detector 12 but since Phase 6 no longer has a	
	call, another activation of Detector 12 will continue to extend Phase	
	4.	
53.	Set HITL Detector Input 12 = On and 4 Off	
54.	DELAY .2 Seconds	
55.	Set HITL Detector Input 12 = Off	
56.	DELAY .2 Seconds	
57.	GET ringStatus.1	
58.	VERIFY that RESPONSE VALUE ringStatus 1 AND 0x07 = 0x01	Pass/Fail
	(xxxxx001 = extension)	
59.	GET phaseStatusGroupPhaseOns.1	

ſ		
60.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 AND 0x08 = 0x08	Pass/Fail
	Note: Verify that extensions are still timing on Phase 4 because even	
	chough call was > 4 seconds later, the absence of call on phase 6	
61	Sot HITL Detector Input 7 - Off	
62	DELAX 2 Seconds	
63	SET vobioloDotoctorEvtond 12 = [currentEvtond]/aluo]	
03.		
	Note: Restore original values.	
Detector 12 extend	Is Phase 12 when Phase 12 Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 12 = On	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 6 = On	
18.	DELAY .2 Seconds	
19.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
20.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X08	
21.	DELAY 1 Second	
22.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
23.	WEND	
	Note: Wait until controller reaches 6+12.	
24.	Set HITL Detector Input 6 = Off	
25.	DELAY .2 Seconds	
26.	GET ringStatus.1	
27.	WHILE ringStatus.1 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
28.	DELAY 1 Second	
29.	GET ringStatus.1	

30.	WEND	
	Note: Wait for Max Out Indication.	
	Ring 1 = 2, 3, 4, 9, 11, 12 , & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
31.	GET phaseStatusGroupPhaseOns.2	
32.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2 AND 0x08 = 0x08	Pass/Fail
	Note: Ensure that Max Out occurred on Phase 12.	
33.	Set HITL Detector Input 12 = Off	
34.	DELAY .2 Seconds	
Teardown		
	PERFORM Detector Teardown – TC020 if not proceeding to another	
	detector operation test case	
	Test Case Results	
Tested By:	Date Tested	Pass/Fail
Test Case Notes:	<notes> The section on "Detector 12 extends Phase 4 until Phase 6 call exis Detector 12 gaps and Phase 6 call continues" is not the equivalent as in TC011 and TC015. This may be due to a timing issue. </notes>	ts and the last steps
Version History:	v1.00 05/09/06 Initial Draft – RDR v1.01 07/05/06 Updated notes – RDR v1.02 07/31/06 Implemented script and proofed – JJ	

Detector 13 Operations

Test Case:	Title:	Detector 13 Operations	
TC013	Description:	Verifies the operation of Detector 13 to call Phase 2 under specific conditions, extend Phase 6 under specific conditions, and extend intervals 1625B, 1635B, 1645B, 1735B, 1745B, 1835B, and 1845B.	
	Constants:		
	Variables:		
	Pass/Fail	The DUT shall pass every verification step include	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detect	tor Operations Setup – TC019 if not already done	
	SO.		
Detector 13 No Ca	II on Phase 2 when	o Overlap B is Green	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus	.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Seco	nd	
4.	GET ringStatus	s.1, ringStatus.2	
5.	WEND		
	Note: Loop until co	ontroller rests somewhere	
6	Sot UITL Detector		
Ο.		input 5 – On	

7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY 2 Seconds	
15.	GET overlapStatusGroupGreens.1	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap B = 5+6+X.	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND $0x02 = 0x02$	
	Note: Verifies that Overlap B = Green.	
18.		
19.	Set HILL Detector Input 13 = On	
20.	DELAY 2 Seconds	
21.	GET phaseStatusGroupVehCalls.1, phaseStatusGroupVehCalls.2	
22.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =	Pass/Fail
	0x00 AND phaseStatusGroupVenCalls.2 = 0x00	
	Noto: Chacks for no call on Dhace 2 or anywhere	
	Set HITL Detector Input 13 - Off	
23.	DELAV 2 Seconds	
Detector 12 No Co	I on Phase 2 when Overlan B is not Green and a Call on Phase 3	
	GET ringStatus 1 ringStatus 2	
	WHILE ringStatus 1 \pm 0x03 AND ringStatus 2 \pm 0x03	
2. 3	DELAV 1 Second	
<u> </u>	CET ringStatus 1 ringStatus 2	
<u>т.</u> 5	WEND	
0.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	

15.	GET overlapStatusGroupGreens.1	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap $B = 5+6+X$.	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND 0x02 = 0x00	
	<i>Note:</i> Verifies that Overlap B = NOT Green.	
18.	END IF	
19.	Set HITL Detector Input 3 = On	
20.	DELAY 2 Seconds	
21.	Set HITL Detector Input 13 = On	
22.	DELAY 2 Seconds	
23.	GET phaseStatusGroupVehCalls.1	
24.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1	Pass/Fail
	AND $0x02 = 0x00$	
	Note: Checks for no call on Phase 2	
25.	Set HITL Detector Input 3 = Off	
26.	DELAY .2 Seconds	
27.	Set HITL Detector Input 13 = Off	
28.	DELAY .2 Seconds	
Detector 13 No Ca	II on Phase 2 when Overlap B is not Green and a Call on Phase 4	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	No (a Maita antil a anta llan an ach an 4 a O	
4.0	Note: Walt until controller reaches 1+8.	
13.	Set HILL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	
	Note overlag Status Oracle Charges is antianal and a OFT may return	
	Note: overlapStatusGroupGreens is optional and a GET may return	
10	a nosuchivame. This also assumes that Overlap $B = 5+6+X$.	
16.		
17.	VERIFY THAT RESPONSE VALUE OVERIAPSTATUSGROUPGREENS.	
	Note: Verifies that Overlap $B = NOT$ Groop	
18	FND IF	
10.		

19.	Set HITL Detector Input 4 = On	
20.	DELAY 2 Seconds	
21.	Set HITL Detector Input 13 = On	
22.	DELAY 2 Seconds	
23	GET phaseStatusGroupVehCalls 1	
20.	VERIEV that RESPONSE VALUE AND $0x02 = 0$	Pass/Fail
27.		1 455/1 41
	Note: Checks for no call on Phase 2	
25	Set HITL Detector Input 13 = Off	
26	DELAY 2 Seconds	
20.	Set HITL Detector Input $4 = Off$	
27.	DELAY 2 Seconds	
Detector 13 calls P	Decription 2 occords	250 /
	GET ringStatus 1 ringStatus 2	
ן. ר	GET IIIgOtatus. 1, IIIgOtatus.2 WHILE ringStatus.1 \pm 0x02 AND ringStatus.2 \pm 0x02	
<u>∠.</u>	DELAV 4 Second	
<u>ی</u>	DELAT I Secoliu	
4. 5		
5.	WEND	
	Note: Loop until controller rests somewhere.	
<u>6.</u>	Set HILL Detector Input 7 = On	
1.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap $B = 5+6+X$.	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND 0x02 = 0x00	
	Note: Verifies that Overlap B = NOT Green.	
18.	END IF	
19.	Set HITL Detector Input 13 = On	
20.	DELAY 2 Seconds	
21.	GET phaseStatusGroupVehCalls.1	
22.	VERIFY that RESPONSE VALUE AND 0x02 = 0x02	Pass/Fail
	Note: Checks for call on Phase 2	
23.	Set HITL Detector Input 13 = Off	
24.	DELAY .2 Seconds	

Detector 13 does n	ot extend Phase 6 when Phase 6 is Green and there are not Calls o	n Phase 7 or
Phase 8		
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 13 = On	
	·	
	<i>Note:</i> To possible extend	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 6 = On	
	Note: To force extensions	
18.	DELAY .2 Seconds	
19.	Set HITL Detector Input 2 = On	
	Note: When opposing call exists	
20.	DELAY .2 Seconds	
21.	Set HITL Detector Input 6 = Off	
	Note: To check whether possible extensions are due to Detector 13.	
22.	DELAY .2 Seconds	
23.	GET ringStatus.2	
24.	WHILE ringStatus.1 AND 0x08 ≠ 0x08 (xxxx1xxx = gap out)	
25.	DELAY 1 Second	
26.	GET ringStatus.2	
27.	WEND	
	Note: Wait for Gap Out indication on Ring 2.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
28.	GET phaseStatusGroupPhaseOns.1	
29.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x20 = 0x20	
	Note: Verify that Gap Out indication occurred on Phase 6.	
30.	Set HITL Detector Input 2 and 13 = Off	
31.	DELAY .2 Seconds	

Detector 13 extend	Is Phase 6 when Phase 6 is Green and there is a Call on Phase 7	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
-		
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 13 = On	
-		
	Note: To possible extend	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 6 = On	
	Note: To force extensions	
18.	DELAY .2 Seconds	
19.	Set HITL Detector Input 7 = On	
	Note: When conflicting call exists and satisfy conditions	
20.	DELAY .2 Seconds	
21.	Set HITL Detector Input 6 = Off	
	Note: To check whether possible extensions are due to Detector 13.	
22.	DELAY .2 Seconds	
23.	GET ringStatus.2	
24.	WHILE ringStatus.2 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
25.	DELAY 1 Second	
26.	GET ringStatus.2	
27.	WEND	
	Note: Wait for Max Out Indication on Ring 2.	
	Ring 1 = 2, 3, 4, 9, 11, & 12	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
28.	GET phaseStatusGroupPhaseOns.1	
29.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x20 = 0x20	
	Note: Verify that Max Out indication occurred on Phase 6.	
30.	Set HITL Detector Input 7 and 13 = Off	
31.	DELAY .2 Seconds	

Detector 13 extend	Is Phase 6 when Phase 6 is Green and there is a call on Phase 8	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
-		
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 13 = On	
-		
	Note: To possible extend	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 6 = On	
	Note: To force extensions	
18.	DELAY .2 Seconds	
19.	Set HITL Detector Input 8 = On	
	Note: When opposing call exists	
20.	DELAY .2 Seconds	
21.	Set HITL Detector Input 6 = Off	
	Note: To check whether possible extensions are due to Detector 13.	
22.	DELAY .2 Seconds	
23.	GET ringStatus.2	
24.	WHILE ringStatus.1 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
25.	DELAY 1 Second	
26.	GET ringStatus.2	
27.	WEND	
	Note: Wait for Max Out Indication on Ring 2.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
28.	GET phaseStatusGroupPhaseOns.1	
29.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x20= 0x20	
	Note: Verify that Max Out indication occurred on Phase 6.	
30.	Set HITL Detector Input 8 and 13 = Off	
31.	DELAY .2 Seconds	

Detector 13 extend	Is Phase 13 when Phase 13 is Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4	GET ringStatus 1, ringStatus 2	
5	WEND	
0.		
	Note: Loop until controller rests somewhere	
6	Sot HITL Detector Input 8 - On	
0. 7	DELAV 2 Secondo	
7.	DELAT .2 Seconds	
<u>ŏ.</u>	GET phaseStatusGroupPhaseOns. 1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 3 = On	
16	DELAY 2 Seconds	
17	Set HITL Detector Input 13 = On	
18	DELAY 2 Seconds	
10.	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
19.	GET phaseolalusGloupFlidseOlis.1, phaseolalusGloupFlidseOlis.2	
20.	while phaseStatusGroupPhaseOns. 1 ≠ 0x01 AND	
	phaseStatusGroupPhaseOns.2 # 0x10	
21.	DELAY 1 Second	
22.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
23.	WEND	
	Note: Wait until controller reaches 1+13.	
24.		
25.	Set HITL Detector Input 3 = Off	
26.	DELAY .2 Seconds	
27.	GET ringStatus.2	
28.	WHILE ringStatus.2 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
29.	DELAY 1 Second	
30.	GET ringStatus.2	
31.	WEND	
•		
	Note: Wait for Max Out Indication on Ring 2	
	Ring $1 = 2 \ 3 \ 4 \ 9 \ 11 \ \& 12$	
	Ring $2 = 15$ 16 5 6 7 8 & 13	
32	GET phaseStatusGroupPhaseOns 2	
22.	VEDIEV that DESDONSE VALUE phaseStatusGroupDhaseOns 2	Dass/Fail
		rass/raii
	Noto: Ensure that Max Out occurred on Phase 12	
<u></u> ∂ 4	Note, Ensure that wax out occurred on Phase 13.	
34.	Set THE Detector input 13 = OII	
35.	DELAY 2 Seconds	

Detector 13 Teardown			
	PERFORM Detector Teardown – TC020 if not proceeding to another		
	detector operation test case		
Test Case Results			
Tested By:	Date Tested		
Test Case Notes:			
Version History:	 v1.00 04/12/06 Initial Draft – RDR v1.01 07/05/06 Deleted strikethroughs, moved initial detector turnoff point, corrected detector number in some cases, and updated notes – RDR v1.02 07/13/06 Implemented script and proofed – JJ 		

Detector 14 Operations

Test Case:	Title:	Detector 14 Operations	
TC014	Description:	Verifies the operation of Detector 14 to call Phase	2 under
		specific conditions, extend Phase 6 under specific	conditions,
		and extend intervals 1625B, 1635B, 1645B, 1735B	, 1745B,
		1835B, and 1845B.	
	Constants:		
	Variables:		
	Pass/Fail	The DUT shall pass every verification step include	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup	•		
	PERFORM Detect	or Operations Setup – TC019 if not already done	
	SO.		
Detector 14 No Ca	II on Phase 2 when	Overlap B is Green	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus.	.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Seco	nd	
4.	GET ringStatus	s.1, ringStatus.2	
5.	WEND		
	Note: Loop until co	ontroller rests somewhere.	
6.	Set HITL Detector	Input 3 = On	
7.	DELAY .2 Second	S	
8.	GET phaseStatus	GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseState	usGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroup	PhaseOns.2 ≠ 0x00	
10.	DELAY 1 Seco	nd	
11.	GET phaseStat	tusGroupPhaseOns.1,	
	phaseStatusGr	oupPhaseOns.2	
12.	WEND		
	Note: Wait until co	ntroller reaches 3+5.	
13.	Set HITL Detector	Input 3 = Off	
14.	DELAY 2 Seconds	3	
15.	GET overlapStatus	sGroupGreens.1	
	Note: overlapStatu	IsGroupGreens is optional and a GET may return	
	a noSuchName. T	his also assumes that Overlap $B = 5+6+X$.	

16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND 0x02 = 0x02	
	<i>Note:</i> Verifies that Overlap B = Green.	
18.	END IF	
19.	Set HITL Detector Input 14 = On	
20.	DELAY 2 Seconds	
21.	GET phaseStatusGroupVehCalls.1, phaseStatusGroupVehCalls.2	
22.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =	Pass/Fail
	0x00 AND phaseStatusGroupVehCalls.2 = 0x00	
	Note: Checks for no call on Phase 2 or anywhere	
23.	Set HITL Detector Input 14 = Off	
24.	DELAY .2 Seconds	
Detector 14 No Ca	II on Phase 2 when Overlap B is not Green and a Call on Phase 3	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap B = 5+6+X.	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND 0x02 = 0x00	
	<i>Note:</i> Verifies that Overlap B = NOT Green.	
18.	END IF	
19.	Set HITL Detector Input 3 = On	
20.	DELAY 2 Seconds	
21.	Set HITL Detector Input 14 = On	
22.	DELAY 2 Seconds	
23.	GET phaseStatusGroupVehCalls.1	

24.	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1	Pass/Fail
	AND $0x02 = 0x00$	
	Note: Checks for no call on Phase 2	
25	Set HITL Detector Input $3 = Off$	
20.	DELAX 2 Seconds	
20.	Set HITL Detector Input 14 - Off	
<u> </u>	DELAX 2 Secondo	
20. Detector 14 No Ca	U on Phase 2 when Overlan B is not Green and a Call on Phase 4	
	GET ringStatus 1 ringStatus 2	
ו. י	WHILE ringStatus $1 \neq 0x03$ AND ringStatus $2 \neq 0x03$	
2.	DELAV 1 Second	
J.	CET ringStatus 1, ringStatus 2	
4. E		
5.	WEND	
	Note: Loop until controller rests somewhere	
6	Set HITL Detector Input 8 = On	
7	DELAY 2 Seconds	
<u> </u>	GET nhaseStatusGrounPhaseOns 1 nhaseStatusGrounPhaseOns 2	
0.	WHILE phaseStatusGroupPhaseOns $1 \neq 0x81 \text{ AND}$	
9.	r_{D}	
10	DELAV 1 Second	
10.	GET nhaseStatusGrounPhaseOns 1	
11.	nhaseStatusGrounPhaseOns 2	
12	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap $B = 5+6+X$.	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND 0x02 = 0x00	
	<i>Note:</i> Verifies that Overlap B = NOT Green.	
18.	END IF	
19.	Set HITL Detector Input 4 = On	
20.	DELAY 2 Seconds	
21.	Set HITL Detector Input 14 = On	
22.	DELAY 2 Seconds	
23.	GET phaseStatusGroupVehCalls.1	
24.	VERIFY that RESPONSE VALUE AND 0x02 = 0	Pass/Fail
	Note: Checks for no call on Phase 2	
25.	Set HITL Detector Input 14 = Off	
26.	DELAY .2 Seconds	
27.	Set HITL Detector Input 4 = Off	
28.	DELAY .2 Seconds	

Detector 14 calls P	Phase 2 when Overlap B is not Green and no Calls on Phase 3 or Ph	ase 4
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	GET overlapStatusGroupGreens.1	
	Note: overlapStatusGroupGreens is optional and a GET may return	
	a noSuchName. This also assumes that Overlap $B = 5+6+X$.	
16.	IF RESPONSE ERROR = noError THEN	
17.	VERIFY that RESPONSE VALUE overlapStatusGroupGreens.1	Pass/Fail
	AND 0x02 = 0x00	
	<i>Note:</i> Verifies that Overlap B = NOT Green	
18.	END IF	
19.	Set HITL Detector Input 14 = On	
20.	DELAY 2 Seconds	
21.	GET phaseStatusGroupVehCalls.1	
22.	VERIFY that RESPONSE VALUE AND 0x02 = 0x02	Pass/Fail
	Note: Checks for call on Phase 2	
23.	Set HITL Detector Input 14 = Off	
24.	DELAY .2 Seconds	
Detector 14 does r	ot extend Phase 6 when Phase 6 is Green and there are not Calls o	n Phase 7 or
Phase 8		
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
-		
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	

11	GET phaseStatusGroupPhaseOns 1	
	nhaseStatusGrounPhaseOns 2	
10		
12.	WEND	
	Nata: Mait until controllar reaches 1+6	
10	Set UITL Detector Input 6 - Off	
IJ.		
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 14 = On	
	Note: To possible extend	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 6 = On	
	Note: To force extensions	
18.	DELAY .2 Seconds	
19.	Set HITL Detector Input 2 = On	
	Note: When opposing call exists	
20.	DELAY .2 Seconds	
21.	Set HITL Detector Input 6 = Off	
	Note: To check whether possible extensions are due to Detector 14.	
22.	DELAY .2 Seconds	
23.	GET ringStatus.2	
24.	WHILE ringStatus.1 AND 0x08 \neq 0x08 (xxxx1xxx = gap out)	
25.	DELAY 1 Second	
26	GET ringStatus 2	
27	WEND	
21.	WEND	
	Note: Wait for Gan Out indication on Ring 2	
	Ring 1 = $2.3 4.0.11 + 12.8 + 1$	
	Ring $2 = 15$ 16 5 6 7 8 & 13	
28	CET nhaseStatusCrounPhaseOne 1	
20.	VEDIEV that DESDONSE VALUE phaseStatusCroupDhaseOne 1	Dace/Eail
29.		Fass/Fall
	AND 0X20 - 0X20	
	Note: Verify that Can Out indication accurred on Phase 6	
20	Note. Verify that Gap Out indication occurred on Phase 6.	
ა <u>ს</u> . ექ		
<u>31.</u>	DELAY .2 Seconds	
Detector 14 extend	IS Phase 6 when Phase 6 IS Green and there IS a Call on Phase 7	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	

11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 14 = On	
	Note: To possible extend	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 6 = On	
	Note: To force extensions	
18.	DELAY .2 Seconds	
19.	Set HITL Detector Input 7 = On	
	Note: When conflicting call exists and satisfies conditions	
20.	DELAY .2 Seconds	
21.	Set HITL Detector Input 6 = Off	
	<i>Note:</i> To check whether possible extensions are due to Detector 14.	
22.	DELAY .2 Seconds	
23.	GET ringStatus.2	
24.	WHILE ringStatus.2 AND $0x10 \neq 0x10$ (xxx1xxxx = maxout)	
25.	DELAY 1 Second	
26.	GET ringStatus.2	
27.	WEND	
	Note: Wait for Max Out Indication on Ring 2.	
	Ring 1 = 2, 3, 4, 9, 11, & 12 $P_{10} = 2, 5, 4, 9, 11, 8, 12$	
	Ring $2 = 15, 16, 5, 6, 7, 8, \& 13$	
28.	GET phaseStatusGroupPhaseOns.1	D / C - 11
29.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND $0x_20 = 0x_20$	
	Nater Varify that May Out indication accurred on Dhase 6	
20	Note. Verily that wax out mulcation occurred on Phase 6.	
21	DELAV 2 Seconda	
Detector 14 extens	DELAT .2 Seconds	
	CET ringStatus 1 ringStatus 2	
<u>າ.</u>	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
2.	DELAV 1 Second	
J.	GET ringStatus 1 ringStatus 2	
<u>т.</u> 5	WEND	
5.	WEND	
	Note: Loop until controller rests somewhere	
6	Set HITI Detector Input 6 = On	
7	DELAY 2 Seconds	
8	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
<u> </u>	WHILE phaseStatusGroupPhaseOns $1 \pm 0x21$ AND	
Ŭ.	phaseStatusGroupPhaseOns.2 \neq 0x00	
10.	DELAY 1 Second	
-		

11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 14 = On	
	Note: To possible extend	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 6 = On	
	Note: To force extensions	
18.	DELAY .2 Seconds	
19.	Set HITL Detector Input 8 = On	
	Note: When opposing call exists	
20.	DELAY 2 Seconds	
21.	Set HIL Detector input 6 = Off	
	Note: To check whether possible extensions are due to Detector 14	
22	DELAX 2 Seconds	
22.	CET ringStatus 2	
23.	WHILE ringStatus 1 AND $0x10 \neq 0x10$ (xxx1xxxx - maxout)	
27.	DELAV 1 Second	
26	GET ringStatus 2	
20.	WEND	
21.		
	Note: Wait for Max Out Indication on Ring 2.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
28.	GET phaseStatusGroupPhaseOns.1	
29.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x20= 0x20	
	<i>Note:</i> Verify that Max Out indication occurred on Phase 6.	
30.	Set HITL Detector Input 8 and 14 = Off	
31.	DELAY .2 Seconds	
Detector 14 extend	Is Phase 13 when Phase 13 is Green	[
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Nota: Loop until controller rests somowhoro	
6	Sat HITI Detector Input $8 = \Omega n$	
7	DELAY 2 Seconds	
<u> </u>	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
۵. ۵	WHILE nhaseStatusGrounPhaseOne 1 \pm 0v81 AND	
υ.	phaseStatusGroupPhaseOns $2 \neq 0x01$ AND	
10.	DELAY 1 Second	

11.	GET phaseStatusGroupPhaseOns.1,		
	phaseStatusGroupPhaseOns.2		
12.	WEND		
	Note: Wait until controller reaches 1+8.		
13.	Set HITL Detector Input 8 = Off		
14.	DELAY .2 Seconds		
15.	Set HITL Detector Input 3 = On		
16.	DELAY .2 Seconds		
17.	Set HITL Detector Input 14 = On		
18.	DELAY .2 Seconds		
19.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2		
20.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x01 AND		
	phaseStatusGroupPhaseOns.2 ≠ 0x10		
21.	DELAY 1 Second		
22.	GET phaseStatusGroupPhaseOns.1,		
	phaseStatusGroupPhaseOns.2		
23.	WEND		
	Note: Wait until controller reaches 1+13.		
24.			
25.	Set HITL Detector Input 3 = Off		
26.	DELAY .2 Seconds		
27.	GET ringStatus.2		
28.	WHILE ringStatus.2 AND $0x10 \neq 0x10$ (xxx1xxxx = maxout)		
29.	DELAY 1 Second		
30.	GET ringStatus 2		
31.	WEND		
011			
	Note: Wait for Max Out Indication on Ring 2.		
	Ring $1 = 2, 3, 4, 9, 11, \& 12$		
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13		
32.	GET phaseStatusGroupPhaseOns.2		
33.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2	Pass/Fail	
	AND 0x10 = 0x10		
	Note: Ensure that Max Out occurred on Phase 13.		
34.	Set HITL Detector Input 14 = Off		
35.	DELAY .2 Seconds		
Detector 14 Teardo	bwn		
	PERFORM Detector Teardown – TC020 if not proceeding to another		
	detector operation test case		
Test Case Results			
	Date		
Tested By:	Tested		
Test Case Notes:	100,00		
Version History	v1.00_04/12/06_Initial Draft – RDR		
• 0131011 1 113101 y.	v1.01.07/05/06 Deleted strikethroughs moved initial detector turnoff r	oint	
	corrected detector number in some cases, and undated notes -	- RDR	
	v1.02.07/17/06. Implemented script and proofed – .I.I		
	1		

Detector 15 Operations

Test Case:	Title:	Detector 15 Operations	
TC015	Description:	Verifies the operation of Detector 15 to call and extends phase 6	
		under specific conditions.	•
	Constants:		
	Variables:		
	Pass/Fail	The DUT shall pass every verification step included	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detector Operations Setup – TC019 if not already done		
	SO.		
Detector 15 calls P	hase 6 during 2+5	Green	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03		
3.	DELAY 1 Seco		
4.	GET ringStatus	.1, ringStatus.2	
5.	WEND		
	Natari ann until an	atrollor rooto comourboro	
	Note: Loop until co	ntroller rests somewhere.	
0.	Set HIL Detector	nput 2 = On	
<u> </u>	DELAY .2 Seconds	5 Desus Dhaas Oras (), is haas Otatus Oras in Dhaas Oras ()	
8.		GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatu	ISGroupPhaseOns.1 ≠ 0x12 AND	
10		PhaseOns.2 ≠ 0X00	
10.			
11.		usGroupPhaseOns.1,	
10		JuprilaseOns.2	
12.	Note: Wait until co	atroller reaches 2+5	
13	Set HITI Detector	In other reactives 2+3.	
10.	DELAY 2 Seconde		
15	Set HITI Detector	Input 15 = On	
10.	DELAY 2 Seconds		
10.	GET nhaseStatus(GrounGreens 1 nhaseStatusGrounGreens 2	
	nhaseStatusGroup	VehCalls 1	
18	WHILE phaseState	ISGroupGreens 1 = 0x12 AND	
10.	phaseStatusGroup	Greens.2 = 0x00 AND	
	phaseStatusGroup	VehCalls.1 AND 0x20 ≠ 0x20	
19.	DELAY 1 Seco	nd	
20.	GET phaseStat	usGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGro	pupVehCalls.1	
21.	WEND		
22.	VERIFY phaseStat	usGroupGreens.1 = 0x12 AND	Pass/Fail
	phaseStatusGroup	Greens.2 = 0x00 AND	
	phaseStatusGroup	VehCalls.1 AND 0x20 = 0x20	
	Note: Ensure that I	Phase 6 registers a Vehicle Call during 2+5 Green.	
23.	Set HITL Detector	Input 15 = Off	
24.	DELAY .2 Seconds	6	

Detector 15 calls P	hase 6 during 3+5 Green		
1.	GET ringStatus.1, ringStatus.2		
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03		
3.	DELAY 1 Second		
4.	GET ringStatus.1, ringStatus.2		
5.	WEND		
0.			
	Note: Loop until controller rests somewhere		
6	Set HITL Detector Input 3 = On		
7	DELAY 2 Seconds		
8	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2		
<u> </u>	WHILE phaseStatusCroupPhaseOns $1 \neq 0x14$ AND		
9.	while phaseolalusoloup haseons. $1 \neq 0.00$		
10	DELAX 1 Second		
10.	DELAT I Secoliu		
11.	GET phaseStatusGroupPhaseOns.1,		
	pnaseStatusGroupPnaseOns.2		
12.	WEND Nata Maitantil andrallar reach as 0.5		
	Note: Walt until controller reaches 3+5.		
13.	Set HIIL Detector Input 3 = Off		
14.	DELAY .2 Seconds		
15.	Set HITL Detector Input 15 = On		
16.	DELAY .2 Seconds		
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,		
	phaseStatusGroupVehCalls.1		
18.	WHILE phaseStatusGroupGreens.1 = 0x14 AND		
	phaseStatusGroupGreens.2 = 0x00 AND		
	phaseStatusGroupVehCalls.1 AND 0x20 ≠ 0x20		
19.	DELAY 1 Second		
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,		
	phaseStatusGroupVehCalls.1		
21.	WEND		
22.	VERIEY phaseStatusGroupGreens 1 = 0x14 AND Pass/Fail		
	phaseStatusGroupGreens 2 = 0x00 AND		
	phaseStatusGroupVehCalls.1 AND 0x20 = 0x20		
	······································		
	<i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 3+5 Green.		
23.	Set HITL Detector Input 15 = Off		
24	DELAY 2 Seconds		
Detector 15 calls P	Phase 6 during 4+5 Green		
1	GET ringStatus 1 ringStatus 2		
2	WHILE ringStatus 1 \pm 0x03 AND ringStatus 2 \pm 0x03		
2. 3	DELAY 1 Second		
J.	CET ringStatus 1 ringStatus 2		
5.	WEND		
	Notor Loop until controller roote computers		
<u> </u>			
0.	Set HILL Detector Input 4 = On		
<u> </u>	DELAY .2 Seconds		
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2		
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND		
	phaseStatusGroupPhaseOns.2 ≠ 0X00		
10.	DELAY 1 Second		
11.	GET phaseStatusGroupPhaseOns.1,		
	phaseStatusGroupPhaseOns.2		

12.	WEND		
	Note: Wait until controller reaches 4+5.		
13.	Set HITL Detector Input 4 = Off		
14.	DELAY .2 Seconds		
15.	Set HITL Detector Input 15 = On		
16.	DELAY 2 Seconds		
17	GET phaseStatusGroupGreens 1 phaseStatusGroupGreens 2		
17.	phaseStatusGroupVehCalls 1		
18	WHILE phaseStatusGroupGreens $1 = 0x18$ AND		
10.	phaseStatusGroupGreens 2 = 0x00 AND		
	phaseStatusGroupVehCalls 1 AND $0x20 \neq 0x20$		
10	DELAY 1 Second		
20	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2		
20.	nhaseStatusGrounVehCalls 1		
21.	VEND		
۲۷.	verify phaseStatusGroupGreens. I = 0x18 AND	Pass/Fall	
	phaseStatusGroupGreens.2 = 0x00 AND		
	phaseStatusGroupVenCalls. I AND 0x20 = 0x20		
	Note: Ensure that Dhave Questisters a Mahida Qall during ALE Quest		
	Note: Ensure that Phase 6 registers a vehicle Call during 4+5 Green.		
23.	Set HILL Detector Input 15 = Off		
24.	DELAY .2 Seconds		
Detector 15 calls P	hase 6 during 5+9 Green		
1.	GET ringStatus.1, ringStatus.2		
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03		
3.	DELAY 1 Second		
4.	GET ringStatus.1, ringStatus.2		
5.	WEND		
	Note: Loop until controller rests in green somewhere.		
6.	Set HITL Detector Input 3 = On		
7.	DELAY .2 Seconds		
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2		
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND		
	phaseStatusGroupPhaseOns.2 ≠ 0x00		
10.	DELAY 1 Second		
11	GET phaseStatusGroupPhaseOns 1		
	phaseStatusGroupPhaseOns.2		
12	WEND		
	Note: Wait until controller reaches 3+5		
13	Set HITL Detector Input 3 = Off		
10.	DELAY 2 Seconds		
15	Set HITL Detector Input 7 = On		
16	DELAV 2 Seconde		
17	CET phaseStatusCroupPhaseOne 1, phaseStatusCroupPhaseOne 2		
17.	GET phaseStatusGloupPhaseOns.1, phaseStatusGloupPhaseOns.2		
18.	while phaseStatusGroupPhaseOns.1 ≠ 0x10 AND		
40			
19.	DELAT I SECOND		
20.	GET phaseStatusGroupPhaseOns.1,		
	pnaseStatusGroupPnaseOns.2		
21.	WEND		
	Note: Wait until controller reaches 5+9.		
22.	Set HITL Detector Input 7 = Off		
23.	DELAY .2 Seconds		
24.	Set HITL Detector Input 15 = On		
---------------------	--	-----------	
25.	DELAY .2 Seconds		
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,		
	phaseStatusGroupVehCalls.1		
27.	WHILE phaseStatusGroupGreens.1 = 0x10 AND		
	phaseStatusGroupGreens.2 = 0x01 AND		
	phaseStatusGroupVehCalls.1 AND 0x20 ≠ 0x20		
28.	DELAY 1 Second		
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,		
	phaseStatusGroupVehCalls.1		
30.	WEND		
31.	VERIFY phaseStatusGroupGreens.1 = 0x10 AND	Pass/Fail	
_	phaseStatusGroupGreens.2 = 0x01 AND		
	phaseStatusGroupVehCalls 1 AND $0x20 = 0x20$		
	Note: Ensure that Phase 6 registers a Vehicle Call during 5+9 Green.		
32.	Set HITL Detector Input 15 = Off		
33.	DELAY .2 Seconds		
Detector 15 calls P	hase 6 during 1+7 Green		
1.	GET ringStatus.1. ringStatus.2		
2.	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03		
3	DFLAY 1 Second		
4	GET ringStatus 1 ringStatus 2		
5	WEND		
0.			
	Note: Loop until controller rests somewhere		
6	Set HITL Detector Input 7 = On		
7	DELAY 2 Seconds		
<u> </u>	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2		
0.	WHILE nhaseStatusGroupPhaseOns 1 \pm 0v/1 AND		
Э.	$r_{\text{maseOfalusOfoup}} = 1 \text{ aseOfalusOfoup} = 1 aseOfalusOfou$		
10	DELAV 1 Second		
11	CET phaseStatusCroupPhaseOne 1		
11.	obaseStatusGroupPhaseOns 2		
10			
12.	Victo: Wait until controller reaches 1+7		
12	Sot HITL Detector Input 7 - Off		
11	DELAV 2 Secondo		
14.	Set HITI Detector Input 15 - On		
10.	DELAV 2 Seconde		
10.	DELAT 2 3500103		
17.	GET priaseotatusGroupGreens.1, priaseotatusGroupGreens.2,		
4.0			
18.	while phaseStatusGroupGreens.1 = 0x41 AND		
	phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVabCalle 1 AND 0x20 ± 0x20		
10	DELAX 4 Second		
19.	DELAT I SECONO		
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,		
~ ~ ~			
21.			
22.	VERIFY phaseStatusGroupGreens.1 = 0x41 AND	Pass/Fall	
	phaseStatusGroupGreens.2 = 0x00 AND		
	phaseStatusGroupvenCalls.1 AND 0x20 = 0x20		
	Notes Ensure that Disease Constitutions a Mathematical Control of the Arrange		
	<i>ivote:</i> Ensure that Phase 6 registers a Vehicle Call during 1+7 Green.		

23.	Set HITL Detector Input 15 = Off	
24.	DELAY .2 Seconds	
Detector 15 calls P	hase 6 during 1+8 Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HILL Detector Input 8 = Off	
14.	DELAY 2 Seconds	
15.	Set HILL Detector Input 15 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
10		
10.	vhile phaseStatusGroupGreens. 1 - 0xo1 AND	
	phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVebCalls 1 AND 0x20 \pm 0x20	
19	DELAY 1 Second	
20	GET phaseStatusGroupGreens 1 phaseStatusGroupGreens 2	
20.	phaseStatusGroupVehCalls 1	
21	WEND	
21.	VERIEY phaseStatusGroupGreens 1 = 0x81 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 = 0x20	
	Note: Ensure that Phase 6 registers a Vehicle Call during 1+8 Green.	
23.	Set HITL Detector Input 15 = Off	
24.	DELAY .2 Seconds	
Detector 15 calls P	hase 6 during 2+16 Green	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
<u> </u>	Set HIL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	pnaseStatusGroupPnaseOns.2 ≠ 0x00	
10.	DELAT I SECOND	

11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x80	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 2+16.	
22.	Set HITL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 15 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x80 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 ≠ 0x20	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x80 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 = 0x20	
	Note: Ensure that Phase 6 registers a Vehicle Call during 2+16	
	Green.	
32.	Cot III Detector langet $4E = Off$	
<u></u>	Set HITE Detector input 15 = OII	
33.	DELAY .2 Seconds	
ی. Detector 15 calls P	DELAY .2 Seconds hase 6 during 2+15 Green	
Detector 15 calls P 1.	DELAY .2 Seconds hase 6 during 2+15 Green GET ringStatus.1, ringStatus.2	
33. Detector 15 calls P 1. 2.	DELAY .2 Seconds DELAY .2 Seconds Chase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
33. Detector 15 calls P 1. 2. 3.	DELAY .2 Seconds DBLAY .2 Seconds Chase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	
33. Detector 15 calls P 1. 2. 3. 4.	Set HTL Detector input 15 = On DELAY .2 Seconds Phase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	
33. Detector 15 calls P 1. 2. 3. 4. 5.	Set HTL Detector input 15 = On DELAY .2 Seconds Phase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
33. Detector 15 calls P 1. 2. 3. 4. 5.	Set HTL Detector input 15 = Off DELAY .2 Seconds Phase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
33. Detector 15 calls P 1. 2. 3. 4. 5.	Set HTL Detector input 15 = On DELAY .2 Seconds Phase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere.	
33. Detector 15 calls P 1. 2. 3. 4. 5. 6.	Set HITL Detector input 15 = OII DELAY .2 Seconds Phase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On	
33. Detector 15 calls P 1. 2. 3. 4. 5. 6. 7.	Set HTL Detector input 15 = Off DELAY .2 Seconds Phase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds	
33. Detector 15 calls P 1. 2. 3. 4. 5. 6. 7. 8.	Set HITL Detector input 15 = On DELAY .2 Seconds Phase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
33. Detector 15 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9.	Set HTL Detector input TS = OIT DELAY .2 Seconds Phase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
33. Detector 15 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9.	Set HTL Detector input 15 = On DELAY .2 Seconds Phase 6 during 2+15 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00	
33. Detector 15 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.	Set HTL Detector input TS = Off DELAY .2 Seconds GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second	
33. Detector 15 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	Set HTL Detector input TS = Off DELAY .2 Seconds GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 7 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2	

12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND	
_	phaseStatusGroupPhaseOns.2 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 2+15.	
22.	Set HITL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 15 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x20 ≠ 0x20	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND	Pass/Fail
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20	Pass/Fail
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20	Pass/Fail
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15	Pass/Fail
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green.	Pass/Fail
31. <u>32.</u>	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off	Pass/Fail
31. 32. 33.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds	Pass/Fail
31. 32. 33. Detector 15 calls F	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Phase 6 during 1+13 Green	Pass/Fail
31. 32. 33. Detector 15 calls F 1.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Phase 6 during 1+13 Green GET ringStatus.1, ringStatus.2	Pass/Fail
31. 32. 33. Detector 15 calls F 1. 2.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds hase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	Pass/Fail
31. 32. 33. Detector 15 calls P 1. 2. 3.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Phase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	Pass/Fail
31. 32. 33. Detector 15 calls P 1. 2. 3. 4.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Phase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	Pass/Fail
31. 32. 33. Detector 15 calls F 1. 2. 3. 4. 5.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds hase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	Pass/Fail
31. 32. 33. Detector 15 calls F 1. 2. 3. 4. 5.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Phase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	Pass/Fail
31. 32. 33. Detector 15 calls F 1. 2. 3. 4. 5.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds hase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 6 = On	Pass/Fail
31. 32. 33. Detector 15 calls P 1. 2. 3. 4. 5. 6. 7	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Thase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds	Pass/Fail
31. 32. 33. Detector 15 calls F 1. 2. 3. 4. 5. 6. 7. 8	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds hase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	Pass/Fail
31. 32. 33. Detector 15 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Phase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	Pass/Fail
31. 32. 33. Detector 15 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Phase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.2 ≠ 0x00	Pass/Fail
31. 32. 33. Detector 15 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Phase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second	Pass/Fail
31. 32. 33. Detector 15 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Phase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1.	Pass/Fail
31. 32. 33. Detector 15 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note:</i> Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Phase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second	Pass/Fail
31. 32. 33. Detector 15 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 Note: Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Thase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	Pass/Fail
31. 32. 33. Detector 15 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 Note: Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds Phase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	Pass/Fail
31. 32. 33. Detector 15 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND phaseStatusGroupGreens.2 = 0x40 AND phaseStatusGroupVehCalls.1 AND 0x20 = 0x20 <i>Note</i> : Ensure that Phase 6 registers a Vehicle Call during 2+15 Green. Set HITL Detector Input 15 = Off DELAY .2 Seconds hase 6 during 1+13 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set HITL Detector Input 6 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 \neq 0x21 AND phaseStatusGroupPhaseOns.2 \neq 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND <i>Note:</i> Wait until controller reaches 1+6. Set HITL Detector Input 6 = Off	Pass/Fail

14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 4 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
18	WHILE phaseStatusGroupPhaseOns 1 ≠ 0x01 AND	
10.	phaseStatusGroupPhaseOns $2 \neq 0x10$	
19	DELAY 1 Second	
20	GET nhaseStatusGrounPhaseOns 1	
20.	nhaseStatusGrounPhaseOns 2	
21	WEND	
۷۱.	Note: Wait until controller reaches 1+13	
22	Set HITL Detector Input $A = Off$	
22.	DELAV 2 Seconds	
23. 	Set HITL Detector Input 15 - On	
24.	DELAX 2 Secondo	
20. 06	DELAT .2 Seconds	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	WILLIE phageStatusCroupCroape 1 = 0v01 AND	
۷۱.	while phasestatusGroupGreens. I = 0x01 AND	
	phaseStatusGroupVebCelle 1 AND 0x20 ± 0x20	
	DELAX 4 Second	
<u> </u>	DELAY I Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
20		
30.		
31.	vERIFY phaseStatusGroupGreens. I = 0x01 AND	Pass/Fall
	phaseStatusGroupGreens.2 = 0x10 AND phaseStatusGroupVehCelle 1 AND 0x20 = 0x20	
	phaseStatusGroupVenCalls. I AND 0x20 = 0x20	
	Note: Ensure that Dhase 6 registers a Vahiala Call during 1, 12	
<u></u>	Set UITL Detector Input 15 - Off	
JZ. 22	DELAV 2 Seconda	
Detector 15 oxtone	DELAT .2 Seconds	
which time Detect	is Fildse o when Fildse o is Green until a gap in Delector 15 activity	occurs at
	CET vehicle Detector Extend 15 and DECORD DESDONSE VALUE	
1.	in [currentExtend]/alue]	
1		
2	SET vehicleDetectorExtend $15 = 40$	
2.	SET vehicleDetectorExtend.15 = 40	
2.	SET vehicleDetectorExtend.15 = 40	
2.	SET vehicleDetectorExtend.15 = 40 <i>Note:</i> Set Detector 15 extend time = 4 seconds. GET ringStatus 1 ringStatus 2	
2. 3.	SET vehicleDetectorExtend.15 = 40 <i>Note:</i> Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
2. 3. 4.	SET vehicleDetectorExtend.15 = 40 <i>Note:</i> Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	
2. 3. 4. 5.	SET vehicleDetectorExtend.15 = 40 <i>Note:</i> Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus 1, ringStatus 2	
2. 3. 4. 5. 6. 7	SET vehicleDetectorExtend.15 = 40 Note: Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 W/END	
2. 3. 4. 5. 6. 7.	SET vehicleDetectorExtend.15 = 40 <i>Note:</i> Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
2. 3. 4. 5. 6. 7.	SET vehicleDetectorExtend.15 = 40 <i>Note:</i> Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere	
2. 3. 4. 5. 6. 7. 8	SET vehicleDetectorExtend.15 = 40 Note: Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 2 = On	
2. 3. 4. 5. 6. 7. 8. 9	SET vehicleDetectorExtend.15 = 40 Note: Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 2 = On DELAY 2 Seconds	
2. 3. 4. 5. 6. 7. 8. 9. 10	SET vehicleDetectorExtend.15 = 40 Note: Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 2 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOps 1, phaseStatusGroupPhaseOps 2	
2. 3. 4. 5. 6. 7. 8. 9. 10.	SET vehicleDetectorExtend.15 = 40 <i>Note:</i> Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 2 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	SET vehicleDetectorExtend.15 = 40 Note: Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 2 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND phaseStatusGroupPhaseOns.2 ≠ 0x00	
2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12	SET vehicleDetectorExtend.15 = 40 Note: Set Detector 15 extend time = 4 seconds. GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 2 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second	

13.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
14.	WEND	
	Note: Wait until controller reaches 2+5.	
15.	Set HITL Detector Input 2= Off	
16.	DELAY .2 Seconds	
17	Set HITL Detector Input 15 = On (To call and then extend Phase 6)	
18	DELAY 2 Seconds	
10.	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
20	WHILE phaseStatusGroupPhaseOne 1 \neq 0x21 AND	
20.	$r_{\text{DaseStatusGroupPhaseOns}} 2 \neq 000$	
01	DELAV 1 Second	
<u> </u>	DELAT I Secoliu	
22.	GET phaseStatusGroupPhaseOns.1,	
	pnaseStatusGroupPnaseOns.2	
23.		
	Note: Wait until controller reaches 1+6.	
24.	Set HITL Detector Input 3 = On (So that Detector 15 will extend	
	Phase 6 when there is an opposing call)	
25.	DELAY .2 Seconds	
26.	GET ringStatus.2	
27.	WHILE ringStatus.2 AND 0x07 ≠ 0x01 (xxxxx001 = extension)	
28.	DELAY .1 Second	
29.	GET ringStatus.2	
30.	WEND	
	<i>Note:</i> Wait for indication that extensions are timing.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15 16 5 6 7 8 & 13	
31	GET phaseStatusGroupPhaseOns 1	
32	VERIEV that RESPONSE VALUE phaseStatusGroupPhaseOns 1	Pass/Fail
02.	AND 0x20 = 0x20	1 400/1 41
	Note: Verify that the extensions are on Phase 6	
23	Set HITL Detector Input 15 = Ω ff	
24	DELAV 3 Seconds	
<u> </u>	DELAT 5 Seconds	
ురి. ఎం	DELAX 2 Occession	
30.	DELAY 2 Seconds	
37.	Set HILL Detector Input 15 = Off	
38.	DELAY 2.8 Seconds	
39.	GET ringStatus.2	
40.	VERIFY that RESPONSE VALUE ringStatus.2 AND 0x07 = 0x01	
	(xxxxx001 = extension)	
	<i>Note:</i> Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6 , 7, 8, & 13	
41.	GET phaseStatusGroupPhaseOns.1	
42.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x20 = 0x20	
	Note: Verify that extensions are still timing on Phase 6 because call	
	was entered < 4 seconds later.	
43	Set HITL Detector Input 15 = On	
Δ <u>Δ</u>	DELAY 2 Seconds	
 Λ5	Sat HITL Detector Input 15 - Off	
+0.		

46.	DELAY 4.8 Seconds	
	Note: Since the time between actuations is now 5 seconds, the timer	
	will gap and therefore disable Detector 15 from putting in any further	
	extensions.	
47.	Set HITL Detector Input 15 = On	
48.	DELAY .2 Seconds	
49.	GET ringStatus.2	
50.	VERIFY that RESPONSE VALUE ringStatus.2 AND 0x07 = 0x04	
	(xxxxx100 = Yellow Change)	
	Nata Varify that the Valley, Change is an Dhase 6 because call was	
	Note: Verify that the reliew Change is on Phase o because call was	
	$\frac{\text{Ping 1} = 2 3 4 \text{ G} 11 12 \text{ g} 1}{\text{Ping 1} = 2 3 4 \text{ G} 11 12 \text{ g} 1}$	
	Ring 2 = 15, 16, 5, 6 , 7, 8, & 13	
51.	GFT phaseStatusGroupPhaseOns.1, ringStatus.2.	
-	phaseStatusGroupVehCalls.1	
52.	WHILE (phaseStatusGroupPhaseOns.1 AND 0x20 = 0x20) AND	
	(ringStatus.2 AND 0x07 = 0x04) AND (phaseStatusGroupVehCalls.1	
	AND 0x20 ≠ 0x20)	
53.	DELAY 1 Second	
54.	GET phaseStatusGroupPhaseOns.1, ringStatus.2,	
	phaseStatusGroupVehCalls.1	
55.		D/E11
50.	VERIFY (phaseStatusGroupPhaseOns. I AND UX20 = UX20) AND (ringStatus 2 AND 0y07 = 0y04) AND (phaseStatusGroupVebCalls 1	Pass/Faii
	(IIIIgStatusStat	
	AND 0X20 - 0X20j	
	Note: Ensure that Phase 6 registers a Vehicle Call during 6 Yellow.	
57.	Set HITL Detector Input 3 = Off and 15 = Off	
58.	DELAY .2 Seconds	
59.	SET vehicleDetectorExtend.15 = [currentExtendValue],	
	Note: Restore original values.	
Teardown		
	PERFORM Detector Teardown – TCU20 if not proceeding to another	
	detector operation test case	
	Tast Casa Pasults	
	Date	
Tested By:	Tested	Pass/Fail
Test Case Notes:	<notes></notes>	
Version History:	v1 00 05/05/06 Initial Draft – RDR	
Voloion micro. y.	v1.01 07/05/06 Updated notes – RDR	
	v1.02 07/12/06 Implemented script and proofed – JJ	

Detector 16 Operations

Test Case:	Title:	Detector 16 Operations	
TC016	Description:	Verifies the operation of Detector 16 to call and extend Phase 8	
	·	under specific conditions and extend interval 1825	3 (2+16).
	Constants:	·	、 ,
	Variables:	currentExtendValue	
	Pass/Fail	The DUT shall pass every verification step included	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup	•		
•	PERFORM Detect	or Operations Setup – TC019 if not already done	
	SO.		
Detector 16 calls F	hase 8 when Phas	e 8 not Green (In 2+16 Green)	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus.	1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Seco	nd	
4.	GET ringStatus	.1, ringStatus.2	
5.	WEND		
	Note: Loop until co	ntroller rests somewhere.	
6.	Set HITL Detector	Input 8 = On	
7.	DELAY .2 Seconds	5	
8.	GET phaseStatus	GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatu	usGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroup	PhaseOns.2 ≠ 0x00	
10.	DELAY 1 Seco	nd	
11.	GET phaseStat	usGroupPhaseOns.1,	
	phaseStatusGr	oupPhaseOns.2	
12.	WEND		
	Note: Wait until co	ntroller reaches 1+8.	
13.	Set HITL Detector	Input 8 = Off	
	DELAY .2 Seconds	5	
15.	Set HITL Detector	Input 2 = On	
16.	DELAY .2 Seconds	5	
17.	GET phaseStatus	GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseState	usGroupPhaseOns.1 ≠ 0x02 AND	
	phaseStatusGroup	PhaseOns.2 ≠ 0x80	
19.	DELAY 1 Seco	nd	
20.	GET phaseStat	usGroupPhaseOns.1,	
	phaseStatusGr	oupPhaseOns.2	
21.	WEND		
~-	Note: Wait until co	ntroller reaches 2+16.	
22.	Set HIL Detector	Input 2 = Off	
23.	DELAY .2 Seconds	3	
24.	Set HITL Detector	Input 16 = On	
25.	DELAY .2 Seconds	8	
26.	GET phaseStatus	GroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroup	VehCalls.1	

27	WHILE phaseStatusGroupGreens 1 = 0x02 AND	
<i>L</i> .	nhaseStatusGrounGreens 2 = 0x80 AND	
	phaseStatusGroupGreens.2 – $0x00$ AND $yy00 \neq 0y000$	
00	pnaseStatusGroupVenCalls.1 AND 0x80 ≠ 0x080	
28.		
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x80 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 = 0x80	
	Note: Ensure that Phase 8 registers a Vehicle Call during 2+16	
	Green	
32	Set HITL Detector Input 16 = Off	
22	DELAV 2 Seconds	
JJ. Detector 16 cello F	DELAT .2 Seconds	
	nase o when Phase o hot Green (in 2+15 Green)	L.
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY_2 Seconds	
8	GET phaseStatusGroupPhaseOns 1 phaseStatusGroupPhaseOns 2	
9	WHILE nhaseStatusGrounPhaseOns $1 \neq 0x41$ AND	
0.	nhaseStatusGroupPhaseOns $2 \neq 0x00$	
10	DELAY 1 Second	
11	GET nhaseStatusGrounPhaseOns 1	
11.	nhaseStatusGrounDhaseOns 2	
10		
12.	WEND	
	Note: Wait until controller reaches 1+7	
13	Set HITL Detector Input $7 = Off$	
11	DELAX 2 Seconde	
14.	Set HITL Detector Input 2 = On	
10.	DELAV 2 Seconda	
10.	DELAT .2 Jeculius	
17.	WILL F here Cletus Group Phase Ons. 1, phase Status Group Phase Ons. 2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND	
	pnaseStatusGroupPnaseOns.2 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 2+15.	
22.	Set HITL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 16 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	

27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 ≠ 0x080	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVenCalls.1 AND 0x80 = 0x80	
	Note: Ensure that Dhase 9 registers a Vahiala Call during 2115	
	Groop	
20	Sot HITL Detector Input 16 - Off	
32.	DELAV 2 Seconds	
Detector 16 calls E	DELAT .2 Seconds Phase 8 when Phase 8 not Green (In 1+13 Green)	
	GET ringStatus 1 ringStatus 2	
່. ົ	WHILE ringStatus 1, migotatus 2 WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
2.	DELAV 1 Second	
J.	CET ringStatus 1 ringStatus 2	
<u>4.</u> Б		
5.	WEND	
	Note: Loop until controller rests somewhere	
6	Set HITL Detector Input $7 = On$	
7	DELAY 2 Seconds	
8	GET nhaseStatusGrounPhaseOns 1_nhaseStatusGrounPhaseOns 2	
<u> </u>	WHILE phaseStatusGroupPhaseOns $1 \neq 0x41$ AND	
0.	phaseStatusGroupPhaseOns $2 \neq 0x00$	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 3 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x01 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x10	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Weit until controller reaches 1, 12	
<u></u>	Sot HITL Detector Input 3 - Off	
<u> </u>	DELAV 2 Seconds	
∠J.	DELAT .2 Jeconus Sat HITL Datastar Input 16 - On	
<u>24</u> . 25	DELAV 2 Seconde	
20.	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
20.	nhaseStatusGroun/ehCalle 1	
l		

27.	WHILE phaseStatusGroupGreens.1 = 0x01 AND phaseStatusGroupGreens.2 = 0x10 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 ≠ 0x080	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x01 AND phaseStatusGroupGreens.2 = 0x10 AND phaseStatusGroupVehCalls.1 AND 0x80 = 0x80	Pass/Fail
	<i>Note:</i> Ensure that Phase 8 registers a Vehicle Call during 1+13 Green.	
32	Set HITL Detector Input 16 = Off	
33	DELAY 2 Seconds	
Detector 16 calls P	Phase 8 when Phase 8 not Green (In 2+5 Green)	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 \pm 0x03 AND ringStatus 2 \pm 0x03	
<u>ک.</u> ۲	DELAV 1 Second	
J.	CET ringStatus 1 ringStatus 2	
<u>т.</u> Б		
5.	WEND	
	Note: Loop until controller rests somewhere	
6	Set HITL Detector Input $2 = On$	
7	DELAY 2 Seconds	
<u> </u>	CET nhaseStatusCrounPhaseOns 1_nhaseStatusCrounPhaseOns 2	
0.	WHII = phaseStatusCroupPhaseOns.1, phaseStatusCroupPhaseOns.2	
9.	while phase status group Phase Ons. $1 \neq 0.12$ AND phase Status Group Phase Ons. $2 \neq 0.200$	
10		
10.	DELAT I Secoliu	
11.	GET phaseStatusGroupPhaseOns.1,	
10		
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13	Set HITL Detector Input 2 = Off	
14	DELAY 2 Seconds	
15	Set HITL Detector Input 16 = On	
16.	DELAY 2 Seconds	
17	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
	nhaseStatusGrounVehCalls 1	
18	WHILE nhaseStatusGrounGreens 1 = 0x12 AND	
10.	phaseStatusGroupGreens 2 = 0x00 AND	
	phaseStatusGroupVehCalls 1 AND $0x80 \neq 0x80$	
19	DELAY 1 Second	
20	GET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
20.	phaseStatusGroupVehCalls 1	
21	WEND	
21.	VERIEV nhaseStatusGrounGreens $1 = 0x12 \text{ AND}$	Pass/Fail
22.	$r_{\rm rescalar} = 0.00$ AND	1 033/1 01
	phaseStatusGroupVehCalls 1 AND 0x80 = 0x80	
	Note: Ensure that Phase 8 registers a Vehicle Call during 2+5 Green.	
23.	Set HITL Detector Input 16 = Off	
24.	DELAY .2 Seconds	

Detector 16 calls F	Phase 8 when Phase 8 not Green (In 3+5 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5	WEND	
0.	TERB .	
	Note: Loop until controller rests somewhere	
6	Set HITL Detector Input 3 = On	
7	DELAV 2 Seconds	
7	CET nhaseStatusCrounPhaseOns 1 nhaseStatusCrounPhaseOns 2	
0.	WHILE phaseStatusGroupPhaseOns 1 + 0x14 AND	
9.	$P_{\text{A}} = P_{\text{A}} $	
10	DELAV 1 Second	
10.	CET phageStatusCroupDhageOne 1	
11.	GET phaseStatusGroupPhaseOns.1,	
10		
12.	WEND	
	Nata: Mait until controllar reaches 3+5	
12	Set UITL Detector Input 2 - Off	
13.	Set HITL Detector input 5 = On	
14.	DELAY .2 Seconds	
15.	Set HILL Detector Input 16 = On	
10.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
40		
18.	while phaseStatusGroupGreens.1 = 0x14 AND	
	phaseStatusGroupVebCelle 1 AND 0x20 4 0x20	
10	DELAX 4 Second	
19.	DELAY I Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
<u> </u>	VEND	Dece/Feil
ΖΖ.	vERIFY phaseStatusGroupGreens. I = 0x14 AND phaseStatusGroupGreens. 2 = 0x00 AND	Pass/Fall
	phaseStatusGroupVabCalle 1 AND 0x20 = 0x20	
	priasestatus Group vencalis. TAND 0x00 – 0x00	
	Noto: Ensure that Phase 8 registers a Vehicle Call during 3+5 Green	
	Set HITL Detector Input 16 - Off	
23.	DELAV 2 Seconds	
Detector 16 calls F	base 8 when Phase 8 not Green (In 4 5 Green)	
	CET ringStatue 1 ringStatue 2	[
۱. ۲	GET IIIIgStatus T, IIIIgStatus Z	
<u> </u>	DELAV 1 Second	
ى. ۸	DELAT I SECUIU	
<u>4.</u> Б		
5.	WEND	
	Note: Loop until controller rests somewhere	
6	Sat HITI Detector Input $4 - \Omega n$	
7	DELAV 2 Seconde	
<i>· · · · · · · · · · · · · · · · · · · </i>	CET nhaseStatusCrounDhaseOne 1, nhaseStatusCrounDhaseOne 2	
0. ^	WHILE phaseStatusCroupPhaseOns.1, phaseStatusCroupPhaseOns.2	
Э.	while phasestatus Group Phase Ons. $1 \neq 0.00$	
40		
10.	DELATISECOND	

11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 16 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
40		
18.	while phaseStatusGroupGreens. I = 0x18 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
10	pnaseStatusGroupVenCalls.1 AND 0x80 ≠ 0x80	
19.	DELAY 1 Second	
20.	GE1 phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 = 0x80	
	r	
	Note: Ensure that Phase 8 registers a Vehicle Call during 4+5 Green.	
23.	Set HITL Detector Input 16 = Off	
24.	DELAY .2 Seconds	
Detector 16 calls F	Phase 8 when Phase 8 not Green (In 6+12 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 4+5.	
13.	Set HITL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1.	
-	phaseStatusGroupPhaseOns.2	

21.	WEND	
	Note: Wait until controller reaches 6+12.	
22.	Set HITL Detector Input 6 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 16 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x08 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 ≠ 0x80	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x08 AND	
	phaseStatusGroupVenCalls.1 AND 0x80 = 0x80	
	Note: Ensure that Dhass Quesisters a Mahiela Call during CudQ	
	Note: Ensure that Phase 8 registers a vehicle Call during 6+12	
	Green.	
<u>32.</u>	Set HIL Detector input 16 = OII	
33. Detector 40 collo D	DELAY .2 Seconds	
	All the second s	
<u>ا.</u>	GET IIIIgStatus. 1, IIIIgStatus.2	
<u> </u>	DELAX 1 Second	
<u>ک</u> .	DELAT I Second	
4. E		
5.	WEND	
	Note: Loop until controller rests in green somewhere	
6	Set HITL Detector Input $3 = \Omega n$	
7	DELAY 2 Seconds	
<u> </u>	CET nhaseStatusCrounPhaseOns 1_nhaseStatusCrounPhaseOns 2	
0.	WHILE phaseStatusGroupPhaseOns 1 + 0v14 AND	
5.	nhaseStatusGrounPhaseOns $2 \neq 0x00$	
10	DELAY 1 Second	
10.	GET phaseStatusGroupPhaseOns 1	
	phaseStatusGroupPhaseOns 2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x04	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	

21.	WEND	
	Note: Wait until controller reaches 6+11.	
22.	Set HITL Detector Input 6 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 16 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 ≠ 0x80	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 = 0x80	
	Note: Ensure that Phase 8 registers a Vehicle Call during 6+11	
	Green.	
32.	Set HIIL Detector Input 16 = Off	
33.	DELAY .2 Seconds	
Detector 16 calls P	hase 8 when Phase 8 not Green (In 5+9 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	No to the second the second state in the second second second	
~	Note: Loop until controller rests in green somewhere.	
<u> </u>	Set HILL Detector Input 3 = On	
1.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
10	DELAX 1 Second	
10.	DELAY I Second	
11.	GET phaseStatusGroupPhaseOns. 1,	
10	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Mait until controller reaches 3+5	
13	Set HITL Detector Input $3 = Off$	
10.	DELAY 2 Seconds	
14.	Set HITL Detector Input 7 - On	
16	DELAY 2 Seconds	
17	CET nhaseStatusGrounPhaseOne 1_nhaseStatusGrounPhaseOne 2	
12	WHILE nhaseStatusGrounDhaseOns $1 \neq 0.010$ AND	
10.	nhaseStatusGroupPhaseOns 2 + 0v01	
10	DELAV 1 Second	
18. 20	CET phaseStatusCroupDhaseOps 1	
20.	nhaseStatusGrounPhaseOns 2	

21.	WEND	
	Note: Wait until controller reaches 5+9.	
22.	Set HITL Detector Input 7 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 16 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x10 AND	
	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 ≠ 0x80	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x10 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVehCalls.1 AND 0x80 = 0x80	
	Note: Ensure that Phase 8 registers a Vehicle Call during 5+9 Green.	
32.	Set HITL Detector Input 16 = Off	
33.	DELAY .2 Seconds	
Detector 16 calls F	Phase 8 when Phase 8 not Green (In 1+6 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
<u> </u>	Set HILL Detector Input 6 = On	
1.	DELAY 2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
4.0	pnaseStatusGroupPnaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
10	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Noto: Wait until controllor reaches 1+6	
12	Note: Walt until controller reaches 1 ± 0 .	
11	DELAX 2 Seconds	
14.	Set HITL Detector Input 16 - On	
10.	DELAX 2 Seconds	
10.	CET nhaseStatusGrounGreens 1 nhaseStatusGrounGreens 2	
17.	nhaseStatusGrounVehCalls 1	
18	WHILE nhaseStatusGrounGreens $1 = 0x21 \Delta ND$	
10.	nhaseStatusGrounGreens $2 = 0x00 \text{ AND}$	
	phaseStatusGroupVehCalls 1 AND $0x80 \pm 0x80$	
10	DELAY 1 Second	
20	GET phaseStatusGroupGreens 1_phaseStatusGroupGreens 2	
20.	phaseStatusGroupVehCalls.1	

21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x21 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x80= 0x80	
	Note: Ensure that Phase 8 registers a Vehicle Call during 1+6 Green.	
23.	Set HITL Detector Input 16 = On	
24.	DELAY .2 Seconds	
Detector 16 calls F	Phase 8 when Phase 8 not Green (In 1+7 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HILL Detector Input 7 = Off	
14.	DELAY 2 Seconds	
15.	Set HILL Detector Input 16 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
10		
18.	while phasestatusGroupGreens. I = 0x4 I AND	
	phaseStatusGroupGreens.z = 0x00 AND phaseStatusGroupVobCalle 1 AND 0x80 + 0x80	
10	DELAV 1 Second	
19. 20	CET phaseStatusGroupGroops 1, phaseStatusGroupGroops 2	
20.	nhaseStatusGrounVehCalls 1	
21	WEND	
21.	VERIEV nhaseStatusGrounGreens 1 = 0x41 AND	Pass/Fail
22.	v = 0.477 AND	1 435/1 41
	phaseStatusGroupVehCalls 1 AND $0x80 = 0x80$	
	<i>Note</i> : Ensure that Phase 8 registers a Vehicle Call during 1+7 Green.	
23.	Set HITL Detector Input 16 = Off	
24.	DELAY .2 Seconds	
Detector 16 extend	Is Phase 8 until Phase 2 call exists and Detector 16 gaps and Phase	e 2 call
continues		
1.	GET vehicleDetectorExtend.16 = [currentExtendValue]	
2.	SET vehicleDetectorExtend.16 = 40	
	<i>Note:</i> Set Detector 16 extend time = 4 seconds so that actuation less	
	than 4 seconds apart keep the phase extending.	

3.	GET ringStatus.1, ringStatus.2	
4.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
5.	DELAY 1 Second	
6	GET ringStatus 1 ringStatus 2	
7	WEND	
1.		
	Note: Loop until controller rests somewhere	
Q	Sot HITL Detector Input 4 - On	
<u> </u>	DELAX 2 Secondo	
9. 10	DELAT .2 Seconds	
10.		
11.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
10	pnaseStatusGroupPnaseOns.2 ≠ 0x00	
12.	DELAY 1 Second	
13.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
14.	WEND	
	<i>Note:</i> Wait until controller reaches 4+5 first so that Detector 16 can	
	call and extend Phase 8.	
15.	Set HITL Detector Input 4 = Off	
16.	DELAY .2 Seconds	
17.	Set HITL Detector Input 16 = On (To call and then extend Phase 8)	
18.	DELAY .2 Seconds	
19.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
20.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
21.	DELAY 1 Second	
22.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
23.	WEND	
	Note: Wait until controller reaches 1+8.	
24.	Set HITL Detector Input 2 = On (Detector 16 will extend Phase 8	
	when there is an opposing call)	
25.	DELAY 2 Seconds	
	Note: This should get Phase 4 to start timing extensions and sets up	
	conditional logic.	
26	GET ringStatus 2	
27	WHILE ringStatus 2 AND $0x07 \neq 0x01$ (xxxxx001 = extension)	
28	DELAY 1 Second	
20.	GET ringStatus 2	
20.	WEND	
50.		
	Note: Wait for indication that extensions are timing	
	Ring $1 = 2 \ 3 \ 4 \ 9 \ 11 \ 12 \ 8 \ 1$	
	Ring $2 = 15$ 16 5 6 7 8 & 13	
21	GET nhaseStatusGrounPhaseOns 1	
32	VERIEV that RESPONSE VALUE nhaseStatusGroupPhaseOns 1	Pass/Fail
32.		F 833/F 811
	Note: Verify that the extensions are on Phase 8	
33	Sat HITL Detector Input $16 - Off$	
ວວ. ວາ	DELAV 2 Secondo	
J 34.	DELAT 3 SECONDS	

35.	Set HITL Detector Input 16 = On			
36.	DELAY .2 Seconds			
37.	Set HITL Detector Input 16 = Off			
38.	DELAY 2.8 Seconds			
39.	GET ringStatus.2			
40.	VERIFY that RESPONSE VALUE ringStatus.2 AND 0x07 = 0x01			
	(xxxxx001 = extension)			
	Note: Ring 1 = 2, 3, 4, 9, 11, 12, & 1			
	Ring $2 = 15$ 16 5 6 7 8 & 13			
41	GET phaseStatusGroupPhaseOns 1			
42	VERIEV that RESPONSE VALUE phaseStatusGroupPhaseOns 1			
	AND 0x80 = 0x80			
	Note: Verify that extensions are still timing on Phase 8 because call			
	was entered < 4 seconds later			
43	Set HITL Detector Input 16 = On			
44	DELAY 2 Seconds			
45	Set HITL Detector Input $16 = Off$			
46	DELAV 2.8 Seconds			
40.	GET ringStatus 2			
47.	VEDIEV that DESDONSE VALUE ringStatue 2 AND $0x07 - 0x01$	Dace/Eail		
40.	verifi tilal RESPONSE VALUE TITYStatus.2 AND 0x07 - 0x01	Fass/Fall		
	(xxxxx00) = extension)			
	N_{oto} : Ding 1 = 2, 3, 4, 0, 11, 12, 8, 1			
	Ding $2 = 15$ 16 5 6 7 9 8 12			
40	CET phase Status Croup Dhase One 1			
49. 50	GET phaseOlalusGloupFildseOlis.	Dece/Fail		
50.		Pass/Fail		
	Note: Verify that extensions are still timing on Dhase 9 headyes call			
	was optored < 4 seconds later			
<u> </u>	Sot UITL Detector Input 16 = On			
51. 52	DELAX 2 Secondo			
52.	DELAT .2 Seconds			
	DELAV 4.9 Seconde			
54.	DELAY 4.8 Seconds			
	Note: Since the time between actuations is now E accords, the timer			
	will gap and therefore disable Detector 16 from putting in any further			
	extensions			
55	Sot HITL Detector Input 16 - On			
55.	DELAX 2 Secondo			
50.	DELAT .2 Seconds			
50	GET IIIIgotatus.2 $VEDIEV$ that DECRONCE VALUE ring Status 2 AND 0x07 \neq 0x01	Dece/Feil		
58.	VERIFY that RESPONSE VALUE IngStatus.2 AND $0.07 \neq 0.001$	Pass/Fail		
	(xxxxxuu1 = extension)			
	Note: Verify that Dhase Q is no longer timing extensions			
	$\begin{array}{c} 1 \text{ Ning } 1 = 2, 3, 4, 3, 11, 12, 0 \\ 1 \text{ Ding } 2 = 15, 16, 5, 6, 7, 8, 8, 13 \\ \end{array}$			
50	$\frac{1}{2} = 15, 10, 5, 0, 7, 0, \alpha = 1$			
<u>୦୪</u> .	GET phaseStatusGroupPhaseOns. 1, phaseStatusGroupVenCalls. 1			
60.	vvniLe (phaseStatusGroupPhaseOns.1 AND 0x80 = 0x80) AND			
	(phaseStatusGroupvenCalls. TAND 0X80 ≠ 0X80)			
b1.	DELAY I SECOND			
62.	GET phaseStatusGroupPhaseOns.1,			
	pnaseStatusGroupvenCalls.1			

63.	WEND	
64.	VERIFY (phaseStatusGroupPhaseOns.1 AND 0x80 = 0x80) AND	Pass/Fail
	(phaseStatusGroupVehCalls.1 AND 0x80 = 0x80)	
	Note: Ensure that Phase 8 registers a Vehicle Call.	
65.	Set HITL Detector Input 2 = Off and 16 = Off	
66.	DELAY .2 Seconds	
	POST-CONDITION The Detector 16 vehicleDetectorExtend is still	
	set to 4 seconds	
Detector 16 extend	Is Phase 8 until Phase 2 call exists and Detector 16 gaps but not if I	Phase 2 call
disappears		
	PRE-CONDITION The Detector 16 vehicleDetectorExtend is still set	
	to 4 seconds	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
÷.		
	Note: Loop until controller rests somewhere.	
6	Set HITL Detector Input 4 = On	
7	DELAY 2 Seconds	
8	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2	
0. Q	WHILE phaseStatusGroupPhaseOns $1 \neq 0x18$ AND	
5.	nhaseStatusGrounPhaseOns $2 \pm 0x00$	
10	DELAV 1 Second	
11	CET phaseStatusCroupPhaseOne 1	
11.	obasoStatusGroupPhasoOns 2	
10		
12.	WEND	
	Note: Wait until controller reaches 4+5	
13	Set HITL Detector Input $A = Off$	
11	DELAV 2 Seconde	
14.	Set $HITL$ Detector input 16 = Op (To call and then extend 1+9)	
10.	DELAX 2 Secondo	
10.	DELAT 2 3600103	
1/.	WILL phaseStatusGroupPhaseOns. I, phaseStatusGroupPhaseOns.2	
Ið.	wniLe phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
10		
19.	DELAY 1 Second	
20.	GEI phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Ivote: wait until controller reaches 1+8.	
22.	Set HILL Detector Input 2 = On	
23.	DELAY 2 Seconds	
	Note: Detector 12 will extend Phase 4 when there is an opposing	
24.	GET ringStatus.2	
25.	WHILE ringStatus.2 AND 0x07 ≠ 0x01 (xxxxx001 = extension)	
26.	DELAY .1 Second	
27.	GET ringStatus.2	

28.	WEND	
	Noto: Wait for indication that avtanciona are timing	
	Ping $1 - 2$ 3 $1 - 2$ 3 $1 - 2$ 3 $1 - 2$ 3 $1 - 2$	
	Ring $2 = 15$ 16 5 6 7 8 & 13	
20	GET nhaseStatusGrounPhaseOns 1	
29.	VEDIEV that DESDONISE VALUE phaseStatusGroupDhaseOne 1	Dace/Eail
30.	AND $0x80 = 0x80$	Fass/Faii
	<i>Note:</i> Verify that the extensions are timing on Phase 8.	
31.	Set HITL Detector Input 16 = Off	
32.	DELAY 3 Seconds	
33.	Set HITL Detector Input 16 = On	
	DELAY .2 Seconds	
35.	Set HITL Detector Input 16 = Off	
36.	DELAY 2.8 Seconds	
37.	GET ringStatus.2	
38.	VERIFY that RESPONSE VALUE ringStatus.2 AND 0x07 = 0x01	Pass/Fail
	(xxxxx001 = extension)	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
39.	GET phaseStatusGroupPhaseOns.1	
40.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x80 = 0x80	
	Note: Verify that extensions are still timing on Phase 8.	
41.	Set HITL Detector Input 16 = On	
42.	DELAY .2 Seconds	
43.	Set HITL Detector Input 16 = Off	
44.	DELAY 2.8 Seconds	
45.	GET ringStatus.2	
46.	VERIFY that RESPONSE VALUE ringStatus.2 AND 0x07 = 0x01	Pass/Fail
	(xxxxx001 = extension)	
	Note: Wait for indication that extensions are timing.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
47.	GET phaseStatusGroupPhaseOns.1	
48.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND $0x80 = 0x80$	
	Note: Verify that extensions are still timing on Phase 8.	
49.	Set HILL Detector Input 16 = On	
50.	DELAY 2 Seconds	
51.	Set HITL Detector Input 16 = Off , 8 = On , 2 = Off and 3 = On	
	<i>Note:</i> Stop extending Phase 8 via Detector 16, turn on Detector 8 to	
	keep Phase 8 extending, turn off Detector 2 opposing demand as	
	part of the logic, and turn on Detector 3 to place opposing demand	
	Detector 8 will keep extending Phase 8 irrespective of Detector 16,	
	no call on Phase 2 will reset Detector 16 gap function, and the call	
	on Phase 3 will enable extensions to continue to time.	

52.	DELAY 4.8 Seconds	
	Note: This would have the effect of allowing the extend timer to gap	
	and therefore disable detector 16 but since Phase 2 no longer has a	
53	o. Set HITL Detector Input 16 = On and 8 Off	
54	DELAY 2 Seconds	
55.	Set HITL Detector Input 16 = Off	
56.	DELAY 2 Seconds	
57.	GET ringStatus.2	
58.	VERIFY that RESPONSE VALUE ringStatus 2 AND 0x07 = 0x01	Pass/Fail
	(xxxxx001 = extension)	
	Note: Wait for indication that extensions are timing.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
59.	GET phaseStatusGroupPhaseOns.1	
60.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x80 = 0x80	
	<i>Note:</i> Verify that extensions are still timing on Phase 8 because even	
	though call was > 4 seconds later, the absence of call on Phase 2	
61	allowed Phase 8 to continue to continue to time extensions.	
<u> </u>	Set HIL Detector input 3 = OII	
62	DELAT .2 Seconds	
05.	SET VehicleDelectorExtend. TO - [currentExtend value]	
	<i>Note:</i> Return Detector 16 extend time back to their original value.	
Detector 16 extend	<i>Note:</i> Return Detector 16 extend time back to their original value.	
Detector 16 extend	<i>Note:</i> Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2	
Detector 16 extend 1. 2.	Note: Return Detector 16 extend time back to their original value. s Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
Detector 16 extend 1. 2. 3.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	
Detector 16 extend 1. 2. 3. 4.	Note: Return Detector 16 extend time back to their original value.Is Phase 16 when Phase 16 GreenGET ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03DELAY 1 SecondGET ringStatus.1, ringStatus.2	
Detector 16 extend 1. 2. 3. 4. 5.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
Detector 16 extend 1. 2. 3. 4. 5.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere.	
Detector 16 extend 1. 2. 3. 4. 5. 6.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9.	Note: Return Detector 16 extend time back to their original value.Is Phase 16 when Phase 16 GreenGET ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03DELAY 1 SecondGET ringStatus.1, ringStatus.2WENDNote: Loop until controller rests somewhere.Set HITL Detector Input 8 = OnDELAY .2 SecondsGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 ANDphaseStatusGroupPhaseOns.2 ≠ 0x00	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.2	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	Note: Return Detector 16 extend time back to their original value.Is Phase 16 when Phase 16 GreenGET ringStatus.1, ringStatus.2WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03DELAY 1 SecondGET ringStatus.1, ringStatus.2WENDNote: Loop until controller rests somewhere.Set HITL Detector Input 8 = OnDELAY .2 SecondsGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 ANDphaseStatusGroupPhaseOns.2 ≠ 0x00DELAY 1 SecondGET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2WEND	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+8.	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+8. Set HITL Detector Input 8 = Off	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+8. Set HITL Detector Input 8 = Off DELAY .2 Seconds	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+8. Set HITL Detector Input 8 = Off DELAY .2 Seconds Set HITL Detector Input 8 = Off DELAY .2 Seconds Set HITL Detector Input 8 = Off	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+8. Set HITL Detector Input 8 = Off DELAY .2 Seconds Set HITL Detector Input 16 = On DELAY .2 Seconds	
Detector 16 extend 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17.	Note: Return Detector 16 extend time back to their original value. Is Phase 16 when Phase 16 Green GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests somewhere. Set HITL Detector Input 8 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 1+8. Set HITL Detector Input 8 = Off DELAY .2 Seconds Set HITL Detector Input 16 = On DELAY .2 Seconds Set HITL Detector Input 16 = On DELAY .2 Seconds Set HITL Detector Input 2 = On	

19.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2			
20.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND			
	phaseStatusGroupPhaseOns.2 ≠ 0x80			
21.	DELAY 1 Second			
22.	GET phaseStatusGroupPhaseOns.1,			
	phaseStatusGroupPhaseOns.2			
23.	WEND			
	Note: Wait until controller reaches 2+16.			
24.	Set HITL Detector Input 2 = Off			
25.	DELAY .2 Seconds			
26.	GET ringStatus.2			
27.	WHILE ringStatus.2 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)			
28.	DELAY 1 Second			
29.	GET ringStatus.2			
30.	WEND			
	Note: Wait for Max Out Indication.			
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1			
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13			
31.	GET phaseStatusGroupPhaseOns.2			
32.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2 Pass/Fail			
	AND 0x80 = 0x80			
	Note: Ensure that Max Out occurred on Phase 16.			
33.	Set HITL Detector Input 16 = Off			
34.	DELAY .2 Seconds			
Teardown				
	PERFORM Detector Teardown – TC020 if not proceeding to another			
	detector operation test case			
	Test Case Results			
- , , p	Date			
Tested By:	Tested			
Test Case Notes:				
Version History:	v1.00 05/09/06 Initial Draft – RDR			
	v1.01 07/05/06 Updated notes – RDR			
	v1.02 07/12/06 Implemented script and proofed – JJ			
	р р р			

Detector 17 Operations

Test Case:	l itle:	Detector 17 Operations	
TC017	Description:	Verifies the operation of Detector 17 to call and extend Phase 3 under specific conditions and extend interval 3516B (6+11).	
	Constants:		`
	Variables:	currentExtendValue	
	Pass/Fail	The DUT shall pass every verification step included	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detect	tor Operations Setup – TC019 if not already done	
	SO.		
Detector 17 calls Phase 3 when Phase 3 not Green (In 2+16 Green)			

1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x0	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HILL Detector Input $\delta = Off$	
14.	DELAY 2 Seconds	
15.	Set HILL Detector Input 2 = On	
16.	DELAY 2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
10	DELAX 1 Second	
19.	DELAY I Second	
20.	GET phaseStatusGroupPhaseOns. 1,	
۷۱.	WEND	
	Note: Wait until controller reaches 2+16	
22	Set HITL Detector Input $2 = Off$	
23	DELAY 2 Seconds	
24.	Set HITL Detector Input 17 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1. phaseStatusGroupGreens.2.	
_	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x80 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x80 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 2+16	
	Green.	
32.	Set HILL Detector input $17 = OII$	
<u>33.</u>	DELAY .2 Seconds	
Detector 17 calls P	nase 3 when Phase 3 hot Green (In 2+15 Green)	
1.	GET HINGSTATUS.T, HINGSTATUS.Z	

2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1.	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY 2 Seconds	
17	GET phaseStatusGroupPhaseOns 1 phaseStatusGroupPhaseOns 2	
18	WHILE phaseStatusGroupPhaseOns $1 \neq 0x02$ AND	
10.	phaseStatusGroupPhaseOns $2 \neq 0x40$	
19	DFLAY 1 Second	
20	GET phaseStatusGroupPhaseOns 1	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 2+15.	
22.	Set HITL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 17 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 2+15	
	Green.	
32.	Set HITL Detector Input 17 = Off	
33.	DELAY .2 Seconds	
Detector 17 calls F	Phase 3 when Phase 3 not Green (In 1+13 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	

3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 3 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x01 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x10	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 1+13.	
22.	Set HITL Detector Input 3 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 17 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x01 AND	
	phaseStatusGroupGreens.2 = 0x10 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x01 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x10 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	Ivote: Ensure that Phase 3 registers a Vehicle Call during 1+13	
20	Gleen. Set HITL Detector Input 17 = Off	
<u>32.</u>	DELAV 2 Seconda	
JJ. Detector 17 colle D	DELAT 2 Seconds	
	CET ringStatus 1 ringStatus 2	
່. ວ	OLT IIIIYOTATUS T, IIIIYOTATUS 2WHILE ringStatus 1 + 0y02 AND ringStatus 2 + 0y02	
<u>∠.</u>	DELAV 1 Second	
J. J.	DELATISECONO	

4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 2 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 17 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x12 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
21. 22.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND	Pass/Fail
21. 22.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND	Pass/Fail
21. 22.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	Pass/Fail
21. 22.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	Pass/Fail
21. 22.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green.	Pass/Fail
21. 22. 23.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off	Pass/Fail
21. 22. 23. 24.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds	Pass/Fail
21. 22. 23. 24. Detector 17 calls F	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green)	Pass/Fail
21. 22. 23. 24. Detector 17 calls F	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1, ringStatus.2	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4.	<pre>WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 Note: Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2</pre>	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5.	<pre>WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 Note: Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND</pre>	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds 'hase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Nate: Loop uptil controllor rosts compatibility</i>	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5.	 WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i>: Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Sat HITL Detector Input 4 = On 	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5. 5.	 WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i>: Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconda 	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5. 6. 7. 8	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOpe 1, phaseStatusGroupPhaseOpe 2	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5. 5. 6. 7. 8. 9	 WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i>: Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns 1 ≠ 0x18 AND 	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds These 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note</i> : Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 \neq 0x18 AND phaseStatusGroupPhaseOns.2 \neq 0x00	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds These 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5. 5. 6. 7. 8. 9. 9. 10.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET naseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.1 ≠ 0x08	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, onter a for the formation of	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11.	 WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i>: Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 	Pass/Fail
21. 22. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12.	 WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i>: Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, wEND 	Pass/Fail
21. 22. 23. 24. Detector 17 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 9. 10. 11. 12.	WEND VERIFY phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupVehCalls.1 AND 0x04 = 0x04 <i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 2+5 Green. Set HITL Detector Input 17 = Off DELAY .2 Seconds Phase 3 when Phase 3 not Green (In 4+5 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 \neq 0x03 AND ringStatus.2 \neq 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 \neq 0x18 AND phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND <i>Note:</i> Wait until controller reaches 4+5.	Pass/Fail

13.	Set HITL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 17 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1. phaseStatusGroupGreens.2.	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens 1 = 0x18 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
_0.	phaseStatusGroupVehCalls.1	
21.	WEND	
22	VERIEY phaseStatusGroupGreens 1 = 0x18 AND	Pass/Fail
<i>LL</i> .	nhaseStatusGroupGreens 2 = 0x00 AND	1 455/1 41
	phaseStatusGroupVehCalls 1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 4+5 Green	
23	Set HITL Detector Input 17 = Off	
20.	DELAY 2 Seconds	
Detector 17 calls E	Phase 3 when Phase 3 not Green (In 6+12 Green)	
	GET ringStatus 1 ringStatus 2	
່. ວ	GET IIIIgOldlus. 1, IIIIgOldlus.2 WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
2.	DELAY 1 Second	
J.	DELAT I Second	
4. <i>F</i>		
Э.	WEND	
	Note: Leen until controller roote computers	
6	Set LUTL Detector Input 4 – On	
0.	DELAY 0 Generate	
1.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
4.0	pnaseStatusGroupPnaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
4.0	Note: Walt until controller reaches 4+5.	
13.	Set HILL Detector Input 4 = Off	
14.	DELAY .2 Seconds	
15.	Set HILL Detector Input 6 = On	
16.	DELAY .2 Seconds	
	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x08	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 6+12.	
22.	Set HITL Detector Input 6 = Off	
23.	DELAY .2 Seconds	

24.	Set HITL Detector Input 17 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x08 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
28.	DELAY 1 Second	
29	GET phaseStatusGroupGreens 1 phaseStatusGroupGreens 2	
-0.	phaseStatusGroupVehCalls 1	
30	WEND	
31	VERIEY phaseStatusGroupGreens 1 = 0x20 AND	Pass/Fail
01.	$r_{\rm aseStatusGroupGreens} 2 = 0x08 \text{ AND}$	r abb/r an
	phaseStatusGroupVehCalls 1 AND $0x04 = 0x04$	
	Note: Ensure that Phase 3 registers a Vehicle Call during 6+12	
	Green	
30	Set HITL Detector Input 17 - Off	
22	DELAV 2 Seconde	
JJ. Detector 17 cello D	DELAT .2 Seconds	
	nase 8 when Phase 8 hot Green (In 6+11 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 7 0x03 AND ringStatus.2 7 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests in green somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns 1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns $2 \neq 0x04$	
19	DFLAY 1 Second	
20	GET nhaseStatusGrounPhaseOns 1	
20.	phaseStatusGroupPhaseOns 2	
21	WEND	
۷۱.		
	Note: Wait until controller reaches 6+11	
22	Set HITI Detector Input $6 = \Omega ff$	
22.	DELAV 2 Seconde	
 	Set HITI Detector Input 17 - On	
۷4.		

25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31	VERIEY phaseStatusGroupGreens 1 = 0x20 AND	Pass/Fail
•	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	<i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 6+11	
	Green.	
32.	Set HITL Detector Input 17 = Off	
33.	DELAY 2 Seconds	
Detector 17 calls F	Phase 3 when Phase 3 not Green (In 5+9 Green)	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 \neq 0x03 AND ringStatus 2 \neq 0x03	
<u> </u>	DELAV 1 Second	
J.	CET ringStatus 1 ringStatus 2	
<u></u> . Б		
J.	WEND	
	Note: Loop until controller rests in green somewhere	
6	Set HITL Detector Input 3 = On	
	DELAV 2 Seconds	
Q 7.	GET nhaseStatusGroupDhaseOns 1, nhaseStatusGroupDhaseOns 2	
0. 0	GET phaseStatusGroupPhaseOns. 1, phaseStatusGroupPhaseOns.2	
9.	while phaseStatusGloupFliaseOlis.1 \neq 0x14 AND phaseStatusCroupDhaseOne 2 \neq 0x00	
10	DELAV 1 Second	
10.	DELAT I SECOLO	
11.	GET phaseStatusGroupPhaseOns.1,	
10	phasestatusGroupPhaseOns.2	
12.	WEND	
	Nata: Wait until controllar reaches 3+5	
10	Note. Walt until controller reaches 3+5.	
10.	DELAX 2 Secondo	
14.	Sat HITL Detector Input 7 - On	
10.	DELAV 2 Seconda	
10.	DELAT .2 Seconds	
17.	GET phaseStatusGroupPhaseOns. T, phaseStatusGroupPhaseOns.2	
18.	while phaseStatusGroupPhaseOns. 1 ≠ 0x10 AND	
10	DELAX 4 Second	
19.	DELAT I Secoliu	
20.	GET phaseStatusGroupPhaseOns.1,	
	phasestatusGroupPhaseOns.2	
21.	WEND	
	Noto: Mait until controller reaches 5 : 0	
	Note, wait until controller reaches 5+9.	
<u> </u>		
23.	DELAT .2 Seconds	
24.	Set HILL Detector Input 17 = On	
25.	DELAY .2 Seconds	

26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x10 AND	
	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x10 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x01 AND	
	phaseStatusGroupVenCalls.1 AND 0x04 = 0x04	
	<i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 5+9 Green	
32	Set HITI Detector Input 17 = Off	
33	DELAY 2 Seconds	
Detector 17 calls P	Phase 3 when Phase 3 not Green (In 1+6 Green)	
1	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 \pm 0x03 AND ringStatus 2 \pm 0x03	
2. 3	DELAV 1 Second	
3. 1	CET ringStatus 1 ringStatus 2	
<u>т.</u> Б	WEND	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 6 = On	
7.	DELAY 2 Seconds	
8.	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+6.	
13.	Set HITL Detector Input 6 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 17 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x21 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x21 AND	Pass/Fail
	pnaseStatusGroupGreens.2 = 0x00 AND	
	pnaseStatusGroupVenCalls.1 AND 0x04= 0x04	
	Note: Ensure that Dhase 2 registers a Vahiala Call during 4 (Correct	
20	Note. Ensure that Phase 3 registers a venicle Call during 1+6 Green.	

24.	DELAY .2 Seconds	
Detector 17 calls P	hase 3 when Phase 3 not Green (In 1+7 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HIL Detector Input / = Off	
14.	DELAY 2 Seconds	
15.	Set HILL Detector Input 17 = On	
10.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
10		
10.	NHILE PHASeSialusGroupGreens. 1 - 0x4 I AND	
	phaseStatusGroupSicens.2 = 0.00 AND phaseStatusGroupVebCalls 1 AND $0.004 \neq 0.004$	
10	DEL AY 1 Second	
20	GET phaseStatusGroupGreens 1 phaseStatusGroupGreens 2	
20.	nhaseStatusGroupVehCalls 1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x41 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 = 0x04	
	Note: Ensure that Phase 3 registers a Vehicle Call during 1+7 Green.	
23.	Set HITL Detector Input 17 = Off	
24.	DELAY .2 Seconds	
Detector 17 calls F	hase 3 when Phase 3 not Green (In 1+8 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
<u> </u>	Set HILL Detector input $\delta = On$	
/.	DELAT .2 Seconds	
Ŏ.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	w⊓iLE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
10	DELAV 1 Second	
10.		

11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 17 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x81 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls 1	
21	WEND	
21.	VERIEV phaseStatusGroupGreens 1 = 0x81 AND	Pass/Fail
22.	nhaseStatusGrounGreens 2 = 0x00 AND	1 433/1 41
	phaseStatusGroupVebCalls 1 AND 0x04 = 0x04	
	<i>Note</i> : Ensure that Phase 3 registers a Vehicle Call during 1+8 Green.	
23.	Set HITL Detector Input 17 = Off	
24.	DELAY .2 Seconds	
Detector 17 extend	is Phase 3 until Phase 6 call exists and Detector 17 gaps and Phase	e 6 call
continues		
1.	GET vehicleDetectorExtend.17 = [currentExtendValue]	
2.	SET vehicleDetectorExtend.17 = 40	
	Note: Set Detector 17 extend time = 4 seconds so that actuations < 4	
	seconds apart keep the phase extending.	
3.	GET ringStatus.1, ringStatus.2	
4.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
5.	DELAY 1 Second	
6.	GET ringStatus.1, ringStatus.2	
7.	WEND	
	Note: Loop until controller rests somewhere.	
8.	Set HITL Detector Input 8 = On	
9.	DELAY .2 Seconds	
10	GET phaseStatusGroupPhaseOns 1 phaseStatusGroupPhaseOns 2	
11	WHILE phaseStatusGroupPhaseOns $1 \neq 0x81$ AND	
	phaseStatusGroupPhaseOns $2 \neq 0x00$	
12	DFLAY 1 Second	
12.	GET nhaseStatusGrounPhaseOns 1	
10.	nhaseStatusGrounPhaseOns 2	
14		
17.	WEND	
	Note: Wait until controller reaches 1+8 first so that Detector 17 can	
	call and extend Phase 3	
15	Set HITL Detector Input 8 = Off	
16	DELAY 2 Seconds	
10.		
17	Set HITL Detector Input 17 = On (To call and then extend Phase 3)	

18.	DELAY .2 Seconds	
19.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
20.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
21.	DELAY 1 Second	
22.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
23.	WEND	
	Note: Wait until controller reaches 3+5.	
24.	Set HITL Detector Input 6 = On (Detector 17 will extend Phase 3	
~~~~	when there is an opposing call)	
25.	DELAY .2 Seconds	
	Note: This should get Dhese 2 to start timing extensions and acts up	
	conditional logic	
26	CET ringStatus 1	
20.	WHILE ringStatus 1 AND $0x07 \neq 0x01$ (xxxxx001 = extension)	
27.	DELAY 1 Second	
20.	GET ringStatus 1	
	WEND	
00.		
	Note: Wait for indication that extensions are timing.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
31.	GET phaseStatusGroupPhaseOns.1	
32.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x04 = 0x04	
	<i>Note:</i> Verify that the extensions are on Phase 3.	
33.	Set HITL Detector Input 17 = Off	
34.	DELAY 3 Seconds	
35.	Set HILL Detector Input 17 = On	
36.	DELAY 2 Seconds	
37.	Set HILL Detector Input 17 = Off	
38.	DELAY 2.8 Seconds	
<u> </u>	GET INDSTATUS.T	Deee/Feil
40.	VERIFY INAL RESPONSE VALUE INGSTATUS. I AND $0.007 = 0.001$	Pass/Fall
	Note: Ring 1 = 2 3 4 9 11 12 & 1	
	Ring $2 = 15, 16, 5, 6, 7, 8, & 13$	
41.	GET phaseStatusGroupPhaseOns.1	
42.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND 0x04 = 0x04	
	Note: Verify that extensions are still timing on Phase 3 because call	
	was entered < 4 seconds later.	
43.	Set HITL Detector Input 17 = On	
44.	DELAY .2 Seconds	
45.	Set HITL Detector Input 17 = Off	
46.	DELAY 2.8 Seconds	
47.	GET ringStatus.1	

48.	VERIFY that RESPONSE VALUE ringStatus.1 AND 0x07 = 0x01 (xxxxx001 = extension)	Pass/Fail
	<i>Note:</i> Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
49.	GET phaseStatusGroupPhaseOns.1	
50.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 AND 0x04 = 0x04	Pass/Fail
	<i>Note:</i> Verify that extensions are still timing on Phase 3 because call was entered < 4 seconds later.	
51.	Set HITL Detector Input 17 = On	
52.	DELAY .2 Seconds	
53.	Set HITL Detector Input 17 = Off	
54.	DELAY 4.8 Seconds	
	<i>Note:</i> Since the time between actuations is now 5 seconds, the timer will gap and therefore disable Detector 17 from putting in any further extensions.	
55.	Set HITL Detector Input 17 = On	
56.	DELAY .2 Seconds	
57.	GET ringStatus.1	
58.	VERIFY that RESPONSE VALUE ringStatus.1 AND 0x07 ≠ 0x01 (xxxxx001 = extension)	Pass/Fail
	<i>Note:</i> Verify that Phase 3 is no longer timing extensions. Ring 1 = 2, 3, 4, 9, 11, 12, & 1 Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
59.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupVehCalls.1	
60.	WHILE (phaseStatusGroupPhaseOns.1 AND 0x04 = 0x04) AND (phaseStatusGroupVehCalls.1 AND 0x04 ≠ 0x04)	
61.	DELAY 1 Second	
62.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupVehCalls.1	
63.	WEND	
64.	VERIFY (phaseStatusGroupPhaseOns.1 AND 0x04 = 0x04) AND (phaseStatusGroupVehCalls.1 AND 0x04 = 0x04)	Pass/Fail
	<i>Note:</i> Ensure that Phase 3 registers a Vehicle Call.	
65.	Set HITL Detector Input 6 = Off and 17 = Off	
66.	DELAY .2 Seconds	
	POST-CONDITION The Detector 17 vehicleDetectorExtend is still	
	set to 4 seconds	
Detector 17 extend	ds Phase 3 until Phase 6 call exists and Detector 17 gaps but not if	Phase 6 call
disappears		
	PRE-CONDITION The Detector 17 vehicleDetectorExtend is still set	
	to 4 seconds	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 8 = On	

7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 17 = On (To call and then extend Phase 3)	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 3+5.	
22.	Set HITL Detector Input 6 = On	
23.	DELAY .2 Seconds	
	<i>Note:</i> The Detector 17 call gets Phase 3 to start timing extensions	
	because of the opposing demand and sets up conditional logic.	
24.	GET ringStatus.1	
25.	WHILE ringStatus.1 AND 0x07 ≠ 0x01 (xxxxx001 = extension)	
26.	DELAY .1 Second	
27.	GET ringStatus.1	
28.	WEND	
	Note: Wait for indication that extensions are timing.	
	Ring $1 = 2, 3, 4, 9, 11, 12, \& 1$	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
29.	GET phaseStatusGroupPhaseOns.1	
30.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND $0x04 = 0x04$	
	Note: Varify that the extensions are timing on Phase 2	
	Set HITL Detector laput 17 = Off	
<u> </u>	DELAV 2 Cocondo	
<u>ు</u> ∠.	DELAT 3 Seconds	
<u> </u>	Set HITL Detector input 17 = On	
34.	DELAY .2 Seconds	
35.	Set HILL Detector input $17 = Off$	
36.	DELAY 2.8 Seconds	
37.	GET ringStatus.1	1
38.	VERIFY that RESPONSE VALUE ringStatus.1 AND 0x07 = 0x01	Pass/Fail
-----------	-----------------------------------------------------------------------	------------
	(xxxxx001 = extension)	
	Note: Wait for indication that extensions are timing	
	Ring $1 = 2$ 3 4 9 11 12 & 1	
	Ring $2 = 15$ 16 5 6 7 8 & 13	
	GET phaseStatusGroupPhaseOns 1	
40	VERIEV that RESPONSE VALUE phaseStatusGroupPhaseOns 1	Pass/Fail
10.	AND $0x04 = 0x04$	1 400/1 41
	<i>Note:</i> Verify that extensions are still timing is on Phase 3.	
41.	Set HITL Detector Input 17 = On	
42.	DELAY .2 Seconds	
43.	Set HITL Detector Input 17 = Off	
44.	DELAY 2.8 Seconds	
45.	GET ringStatus.1	
46.	VERIFY that RESPONSE VALUE ringStatus.1 AND 0x07 = 0x01	Pass/Fail
	(xxxxx001 = extension)	
	Note: Wait for indication that extensions are timing.	
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
47.	GET phaseStatusGroupPhaseOns.1	
48.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND $0x04 = 0x04$	
	Note: Varify that automaiana are still timing on Dhase 2	
40	Note, venity that extensions are still timing on Phase 5.	
<u> </u>	DELAX 2 Seconds	
51	Set HITL Detector Input $17 - Off 3 - On 6 - Off and 7 - On$	
51.	Set TITE Detector input $T = OII, S = OII, 0 = OII and T = OII$	
	Note: Stop extending Phase 3 via Detector 17, turn on Detector 3 to	
	keep Phase 3 extending, turn off Detector 6 opposing demand as	
	part of the logic, and turn on Detector 7 to place opposing demand.	
	Detector 3 will keep extending Phase 3 irrespective of Detector 17,	
	no call on Phase 6 will reset Detector 17 gap function, and the call	
	on Phase 3 will enable extensions to continue to time.	
52.	DELAY 4.8 Seconds	
	Note: This would have the effect of allowing the extend timer to gap	
	and therefore disable Detector 17, but since Phase 6 no longer has a	
	call, another activation of Detector 17 will continue to extend Phase	
E0	3. Det HITI Detector brock 47 - On and 0.06	
53.	Set HILL Detector Input 17 = On and 3 Off	
54. 55	DELAY .2 Seconds	
50. 56	DELAV 2 Seconda	
57. 57	DELAT .2 Seconds	
59	VEDIEV that DESDONSE VALUE ringStatus 1 AND 0x07 - 0x01	Pace/Eail
50.	VERIFY IIIdL RESPONSE VALUE IIIgSIdlus. I AND $0.007 - 0.001$	FdS5/Fdll
	Note: Wait for indication that extensions are timing	
	Ring $1 = 2, 3, 4, 9, 11, 12, \& 1$	
	Ring $2 = 15, 16, 5, 6, 7, 8, \& 13$	
59.	GET phaseStatusGroupPhaseOns.1	

60.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail
	AND $0x04 = 0x04$	
	Note: Verify that extensions are still timing on Phase 3 because even	
	though call was $> 4$ seconds later, the absence of call on Phase 6	
	allowed Phase 3 to continue to time extensions.	
61.	Set HITL Detector Input 7 = Off	
62.	DELAY .2 Seconds	
63.	SET vehicleDetectorExtend.17 = [currentExtendValue]	
	Note: Return Detector 17 extend time back to its original value	
Detector 17 extend	Is Phase 11 when Phase 11 Green	
1.	GET ringStatus.1. ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	No (a) Mait until controller records a 2 / 5	
10	Note: Walt until controller reaches 3+5.	
13.	Set HILL Detector input $3 = Off$	
14.	DELAY .2 Seconds	
15.	Set HILL Detector input 17 = On	
16.	DELAY .2 Seconds	
17.	Set HILL Detector Input 6 = On	
18.	DELAY .2 Seconds	
19.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
20.	while phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
21.	CET phaseStatueGroupPhaseOns 1	
22.	nhaseStatusGrounPhaseOns 2	
23	WEND	
20.	WEND	
	Note: Wait until controller reaches 6+11.	
24.	Set HITL Detector Input 6 = Off	
25.	DELAY .2 Seconds	
26.	GET ringStatus.1	
27.	WHILE ringStatus.1 AND 0x10 ≠ 0x10 (xxx1xxxx = maxout)	
28.	DELAY 1 Second	
29.	GET ringStatus.1	

30.	WEND	
	<i>Note:</i> Wait for Max Out Indication. Ring 1 = 2, 3, 4, 9, 11, 12, & 1	
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13	
31.	GET phaseStatusGroupPhaseOns.2	
32.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.2 AND 0x04 = 0x04	Pass/Fail
	Note: Ensure that Max Out occurred on Phase 11.	
33.	Set HITL Detector Input 17 = Off	
34.	DELAY .2 Seconds	
Teardown		
	PERFORM Detector Teardown – TC020 if not proceeding to another detector operation test case	
	Test Case Results	
Tested By:	Date Tested	
Test Case Notes:		
Version History:	v1.00 05/09/06 Initial Draft – RDR v1.01 07/05/06 Updated notes – RDR v1.02 08/01/06 Implemented script and proofed – JJ	

## Detector 18 Operations

$\mathbf{T}$	<b>-</b> <i>xu</i>		
Test Case:	Title: Detector 18 Operations		
TC018	Description:	ription: Verifies the operation of Detector 18 to call and extend Phase 7	
		under specific conditions and extend interval 1725	B (2+15).
	Constants:		
	Variables:	currentExtendValue	
	Pass/Fail	The DUT shall pass every verification step included	d within the
	Criteria:	Test Case in order to pass the Test Case.	
Test Step	Test Procedure		Results
Number			
Setup			
	PERFORM Detect	or Operations Setup – TC019 if not already done	
	SO.		
Detector 18 calls F	Phase 7 when Phas	e 7 not Green (In 1+8 Green)	
1.	GET ringStatus.1,	ringStatus.2	
2.	WHILE ringStatus.	1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Seco	nd	
4.	GET ringStatus	s.1, ringStatus.2	
5.	WEND		
	Note: Loop until co	ontroller rests somewhere.	
6.	Set HITL Detector	Input 8 = On	
7.	DELAY .2 Seconds	S	
8.	GET phaseStatus	GroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatu	usGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroup	PhaseOns.2 ≠ 0x00	
10.	DELAY 1 Seco	nd	
11.	GET phaseStat	usGroupPhaseOns.1,	
	phaseStatusGr	oupPhaseOns.2	

12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 18 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x81 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x81 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
	<i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 1+8 Green.	
23.	Set HITL Detector Input 18 = Off	
24.	DELAY .2 Seconds	
Detector 18 calls F	Phase 7 when Phase 7 not Green (In 2+16 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 8 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x0	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+8.	
13.	Set HITL Detector Input 8 = Off	
	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x80	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	

21.	WEND	
	Note: Wait until controller reaches 2+16.	
22.	Set HIL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 18 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
~7	phaseStatusGroupVenCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x80 AND	
	pnaseStatusGroupVenCalls.1 AND 0x40 ≠ 0x40	
28	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVenGalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fall
	phaseStatusGroupVebCelle 1 AND 0x40 = 0x40	
	phaseStatusGroupVenCalls. TAND 0x40 = 0x40	
	Noto: Ensure that Phase 7 registers a Vehicle Call during 2+16	
	Green	
20	Set HITL Detector Input 18 - Off	
32.	DELAV 2 Seconds	
Detector 18 calls F	Decar .2 Seconds Dase 7 when Phase 7 not Green (In 2+15 Green)	
	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
<u>ک.</u> ۲	DELAY 1 Second	
0. 	GET ringStatus 1 ringStatus 2	
5	WEND	
0.		
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
-	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 2 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x02 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	

21.	WEND	
	Note: Wait until controller reaches 2+15.	
22.	Set HIL Detector Input 2 = Off	
23.	DELAY .2 Seconds	
24.	Set HILL Detector Input 18 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x02 AND	
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVenCalls.1 AND 0x40 ≠ 0x40	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVenCalls.1	
30.		D/E-1
31.	VERIFY phaseStatusGroupGreens.1 = 0x02 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x40 AND	
	phaseStatusGroupVenCalls. TAND 0x40 = 0x40	
	Noto: Ensure that Phase 7 registers a Vehicle Call during 2+15	
	Green	
20	Sat HITL Detector Input 18 - Off	
22.	DELAV 2 Seconds	
Detector 18 calls F	Decar .2 Seconds Dase 7 when Phase 7 not Green (In 1+13 Green)	
	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 $\pm$ 0x03 AND ringStatus 2 $\pm$ 0x03	
3	DELAY 1 Second	
4	GET ringStatus 1 ringStatus 2	
5	WEND	
0.		
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 7 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 1+7.	
13.	Set HITL Detector Input 7 = Off	
	DELAY .2 Seconds	
15.	Set HITL Detector Input 3 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x01 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x10	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	pnaseStatusGroupPhaseOns.2	

21.	WEND	
	Note: Wait until controller reaches 1+13.	
22.	Set HITL Detector Input 3 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 18 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x01 AND phaseStatusGroupGreens.2 = 0x10 AND phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x01 AND phaseStatusGroupGreens.2 = 0x10 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note:</i> Ensure that Phase 7 registers a Vehicle Call during 1+13 Green.	Pass/Fail
32.	Set HITL Detector Input 18 = Off	
33.	DELAY .2 Seconds	
Detector 18 calls F	Phase 7 when Phase 7 not Green (In 2+5 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 2 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 2+5.	
13.	Set HITL Detector Input 2 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 18 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens 2	
10	phaseStatusGroupVehCalls.1	
18.	phaseStatusGroupGreens.1 = 0x12 AND phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
19.	DELAY 1 Second	

20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x12 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
	Note: Ensure that Phase 7 registers a Vehicle Call during 2+5 Green.	
23.	Set HITL Detector Input 18 = Off	
24.	DELAY .2 Seconds	
Detector 18 calls F	Phase 7 when Phase 7 not Green (In 3+5 Green)	
1.	GET ringStatus.1, ringStatus.2	
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 3 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note: Wait until controller reaches 3+5.	
13.	Set HITL Detector Input 3 = Off	
14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 18 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x14 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVenCalls.1 AND 0x40 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVenGalls.1	
<u> </u>	WEND	Deee/Eeil
ΖΖ.	VERIFY phaseStatusGroupGreens.1 = 0x14 AND	Pass/Fall
	phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVabCalle 1 AND 0x40 = 0x40	
	phaseStatusGroupVenCalls. TAND 0x40 = 0x40	
	Noto: Ensure that Phase 7 registers a Vehicle Call during 3+5 Green	
22	Sat HITI. Detector Input $18 = Off$	
23.	DELAY 2 Seconds	
Detector 18 calls E	Decar .2 0000003	
	GET ringStatus 1 ringStatus 2	
2	WHILE ringStatus 1 $\pm$ 0y03 AND ringStatus 2 $\pm$ 0y03	
<u>د.</u> ع	DEL AV 1 Second	
J	GET ringStatus 1 ringStatus 2	

5.	WEND	
	Note: Loop until controller rests somewhere.	
6.	Set HITL Detector Input 4 = On	
7.	DELAY .2 Seconds	
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x00	
10.	DELAY 1 Second	
11.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
12.	WEND	
	Note Moit until controller reaches 4 5	
10	Note: Walt until controller reaches 4+5.	
13.	Set HITL Detector input 4 = OII	
14.	DELAT .2 Seconds	
10.	Set HITL Detector input 18 = On	
10.	DELAY .2 Seconds	
17.	phaseStatusGroupVehCalls.1	
18.	WHILE phaseStatusGroupGreens.1 = 0x18 AND	
	phaseStatusGroupGreens.2 = 0x00 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2, phaseStatusGroupVehCalls.1	
21.	WEND	
22.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND	Pass/Fail
22.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND	Pass/Fail
22.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	Pass/Fail
22.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	Pass/Fail
22.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green.	Pass/Fail
22.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off	Pass/Fail
22. 23. 24.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds	Pass/Fail
22. 23. 24. Detector 18 calls F	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds Phase 7 when Phase 7 not Green (In 6+12 Green)	Pass/Fail
22. 23. 24. Detector 18 calls F 1.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2	Pass/Fail
22. 23. 24. Detector 18 calls F 1. 2.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	Pass/Fail
22. 23. 24. Detector 18 calls F 1. 2. 3.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds Phase 7 when Phase 7 not Green (In 6+12 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds Phase 7 when Phase 7 not Green (In 6+12 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere.	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5. 6. 7.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5. 6. 7. 8.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5. 6. 7. 8. 9.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.1 ≠ 0x08	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5. 5. 6. 7. 8. 9.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Thase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>hase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET nhaseStatusGroupPhaseOns.1 ≠ 0x18 AND	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, not a to the formation of the top top the top top the top top the top top top the top	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Thase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00	Pass/Fail
22. 23. 24. <b>Detector 18 calls F</b> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Phase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	Pass/Fail
22. 23. 24. Detector 18 calls F 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13.	VERIFY phaseStatusGroupGreens.1 = 0x18 AND phaseStatusGroupGreens.2 = 0x00 AND phaseStatusGroupVehCalls.1 AND 0x40 = 0x40 <i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 4+5 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>hase 7 when Phase 7 not Green (In 6+12 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests somewhere. Set HITL Detector Input 4 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 $\neq$ 0x18 AND phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	Pass/Fail

14.	DELAY .2 Seconds	
15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
18	WHILE phaseStatusGroupPhaseOns $1 \neq 0x20$ AND	
10.	nhaseStatusGroupPhaseOns $2 \pm 0x08$	
10	DELAV 1 Second	
20	CET phaseStatusCroupPhaseOne 1	
20.	of the phase of the phase of the second se	
۷۱.	WEIND	
	Noto: Mait until controller reaches 6+12	
22	Set HITL Detector Input 6 - Off	
22.	DELAX 2 Secondo	
ZJ	DELAT .2 Seconds	
24.	Set HITL Detector input 18 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x08 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x08 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
	<i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 6+12	
	<i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 6+12 Green.	
32.	<i>Note</i> : Ensure that Phase 7 registers a Vehicle Call during 6+12 Green. Set HITL Detector Input 18 = Off	
<u> </u>	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds	
32. 33. Detector 18 calls P	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds Phase 7 when Phase 7 not Green (In 6+11 Green)	
32. 33. Detector 18 calls P 1.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds Phase 7 when Phase 7 not Green (In 6+11 Green) GET ringStatus.1, ringStatus.2	
32. 33. Detector 18 calls P 1. 2.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds Phase 7 when Phase 7 not Green (In 6+11 Green) GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
32. 33. <b>Detector 18 calls P</b> 1. 2. 3.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds <b>hase 7 when Phase 7 not Green (In 6+11 Green)</b> GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second	
32. 33. <b>Detector 18 calls F</b> 1. 2. 3. 4.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>hase 7 when Phase 7 not Green (In 6+11 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	
32. 33. <b>Detector 18 calls P</b> 1. 2. 3. 4. 5.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds         Phase 7 when Phase 7 not Green (In 6+11 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2	
32. 33. <b>Detector 18 calls P</b> 1. 2. 3. 4. 5.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds <b>Thase 7 when Phase 7 not Green (In 6+11 Green)</b> GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND	
32. 33. <b>Detector 18 calls P</b> 1. 2. 3. 4. 5.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds         Phase 7 when Phase 7 not Green (In 6+11 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.	
32. 33. <b>Detector 18 calls P</b> 1. 2. 3. 4. 5.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds         Phase 7 when Phase 7 not Green (In 6+11 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 3 = On	
32. 33. Detector 18 calls P 1. 2. 3. 4. 5. 6. 7	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds         Phase 7 when Phase 7 not Green (In 6+11 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 3 = On         DELAY .2 Seconds	
32. 33. Detector 18 calls P 1. 2. 3. 4. 5. 6. 7. 8	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds <b>'hase 7 when Phase 7 not Green (In 6+11 Green)</b> GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 3 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
32. 33. Detector 18 calls P 1. 2. 3. 4. 5. 5. 6. 7. 8. 9	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds         hase 7 when Phase 7 not Green (In 6+11 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 3 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns 1 ≠ 0x14 AND	
32. 33. Detector 18 calls P 1. 2. 3. 4. 5. 5. 6. 7. 8. 9.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>Thase 7 when Phase 7 not Green (In 6+11 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 $\neq$ 0x14 AND phaseStatusGroupPhaseOns.2 $\neq$ 0x00	
32. 33. Detector 18 calls P 1. 2. 3. 4. 5. 5. 6. 7. 8. 9.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds <b>'hase 7 when Phase 7 not Green (In 6+11 Green)</b> GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 3 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second	
32. 33. Detector 18 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds <b>'hase 7 when Phase 7 not Green (In 6+11 Green)</b> GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 3 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET nbaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
32. 33. Detector 18 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds         hase 7 when Phase 7 not Green (In 6+11 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 3 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2 ≠ 0x00	
32. 33. Detector 18 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds         hase 7 when Phase 7 not Green (In 6+11 Green)         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 3 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
32. 33. Detector 18 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 11. 12.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>hase 7 when Phase 7 not Green (In 6+11 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	
32. 33. Detector 18 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 11. 12.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12 Green. Set HITL Detector Input 18 = Off DELAY .2 Seconds <b>hase 7 when Phase 7 not Green (In 6+11 Green)</b> GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set HITL Detector Input 3 = On DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	
32. 33. Detector 18 calls P 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds <b>hase 7 when Phase 7 not Green (In 6+11 Green)</b> GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 3 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, wence         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, wence         WEND         Note: Wait until controller reaches 3+5.         Set HITL Detector Input 2 = Off	
32. 33. <b>Detector 18 calls P</b> 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14.	Note: Ensure that Phase 7 registers a Vehicle Call during 6+12         Green.         Set HITL Detector Input 18 = Off         DELAY .2 Seconds <b>hase 7 when Phase 7 not Green (In 6+11 Green)</b> GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set HITL Detector Input 3 = On         DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 3+5.         Set HITL Detector Input 3 = Off         DELAY 0 Seconde         GET phaseStatusGroupPhaseOns.2	

15.	Set HITL Detector Input 6 = On	
16.	DELAY .2 Seconds	
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0x04	
19.	DELAY 1 Second	
20.	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
21.	WEND	
	Note: Wait until controller reaches 6+11.	
22.	Set HITL Detector Input 6 = Off	
23.	DELAY .2 Seconds	
24.	Set HITL Detector Input 18 = On	
25.	DELAY .2 Seconds	
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
27.	WHILE phaseStatusGroupGreens.1 = 0x20 AND	
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40	
28.	DELAY 1 Second	
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,	
	phaseStatusGroupVehCalls.1	
30.	WEND	
31.	VERIFY phaseStatusGroupGreens.1 = 0x20 AND	Pass/Fail
	phaseStatusGroupGreens.2 = 0x04 AND	
	phaseStatusGroupVehCalls.1 AND 0x40 = 0x40	
	Note: Ensure that Phase 7 registers a Vehicle Call during 6+11	
	Green.	
32.	DELAV 2 Casanda	
JJ. Detector 19 collo F	DELAT .2 Seconds	
	CET ring Status 1, ring Status 2	
l.	GET IngStatus. 1, IngStatus.2	
<u> </u>	DELAX 4 Concerd	
3.	DELAY 1 Second	
4.	GET ringStatus.1, ringStatus.2	
5.	WEND	
	Note: Loop until controller roots in groop comowhere	
6	Sot HITL Detector Input 3 = On	
7		
<u> </u>	NEL AV 7 Seconde	
υ.	DELAY .2 Seconds GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	
<b>0</b>	DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns 1 ≠ 0x14 AND	
9.	DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns 2 ≠ 0x00	
9.	DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second	
9. 	DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns 1	
9. 10. 11.	DELAY .2 Seconds GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND phaseStatusGroupPhaseOns.2 ≠ 0x00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
9. 10. 11. 12	DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2	
9. <u>10.</u> 11. 12.	DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2	
9. <u>10.</u> 11. 12.	DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 3+5.	
9. <u>10.</u> 11. <u>12.</u> <u>13.</u>	DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 3+5.         Set HITL Detector Input 3 = Off	
9. 10. 11. 12. 13. 14.	DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 3+5.         Set HITL Detector Input 3 = Off         DELAY .2 Seconds	
9. 10. 11. 12. 13. 14. 15.	DELAY .2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 3+5.         Set HITL Detector Input 3 = Off         DELAY .2 Seconds         Set HITL Detector Input 7 = On	

16.	DELAY .2 Seconds			
17.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2			
18.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x10 AND			
	phaseStatusGroupPhaseOns.2 ≠ 0x01			
19.	DELAY 1 Second			
20.	GET phaseStatusGroupPhaseOns.1,			
	phaseStatusGroupPhaseOns.2			
21.	WEND			
	Note: Wait until controller reaches 5+9.			
22.	Set HITL Detector Input 7 = Off			
23.	DELAY .2 Seconds			
24.	Set HITL Detector Input 18 = On			
25.	DELAY .2 Seconds			
26.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,			
	phaseStatusGroupVehCalls.1			
27.	WHILE phaseStatusGroupGreens.1 = 0x10 AND			
	phaseStatusGroupGreens.2 = 0x01 AND			
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40			
28.	DELAY 1 Second			
29.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,			
	phaseStatusGroupVehCalls.1			
30.	WEND			
31.	VERIFY phaseStatusGroupGreens.1 = 0x10 AND	Pass/Fail		
	phaseStatusGroupGreens.2 = 0x01 AND			
	phaseStatusGroupVenCalls.1 AND 0x40 = 0x40			
	Note: Ensure that Dhoos 7 registers a Vahiala Call during ELO Organ			
20	Note: Ensure that Phase 7 registers a vehicle Call during 5+9 Green.			
ు <u>८</u> .	DELAV 2 Seconda			
JJ. Detector 19 collo F	DELAT .2 Seconds			
	CET ringStatue 1 ringStatue 2			
່. ວ	GET IIIIyStatus 1, IIIIyStatus 2 WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03			
2.	DELAV 1 Second			
J.	CET ringStatus 1 ringStatus 2			
<u>4.</u> Б				
5.	WEND			
	Note: Loop until controller rests somewhere			
6	Set HITL Detector Input 6 = On			
7	DELAY 2 Seconds			
8	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2			
9	WHILE phaseStatusGroupPhaseOns $1 \neq 0x21$ AND			
0.	phaseStatusGroupPhaseOns 2 ≠ 0x00			
10.	DELAY 1 Second			
11.	GET phaseStatusGroupPhaseOns.1.			
	phaseStatusGroupPhaseOns.2			
12.	WEND			
	Note: Wait until controller reaches 1+6.			
13.	<i>Note:</i> Wait until controller reaches 1+6. Set HITL Detector Input 6 = Off			
13. 14.	<i>Note:</i> Wait until controller reaches 1+6. Set HITL Detector Input 6 = Off DELAY .2 Seconds			
13. 14. 15.	Note: Wait until controller reaches 1+6. Set HITL Detector Input 6 = Off DELAY .2 Seconds Set HITL Detector Input 18 = On			
13. 14. 15. 16.	Note: Wait until controller reaches 1+6. Set HITL Detector Input 6 = Off DELAY .2 Seconds Set HITL Detector Input 18 = On DELAY .2 Seconds			

17.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,			
	phaseStatusGroupVehCalls.1			
18.	WHILE phaseStatusGroupGreens.1 = 0x21 AND			
	phaseStatusGroupGreens.2 = 0x00 AND			
	phaseStatusGroupVehCalls.1 AND 0x40 ≠ 0x40			
19.	DELAY 1 Second			
20.	GET phaseStatusGroupGreens.1, phaseStatusGroupGreens.2,			
01				
<u> </u>	IENU IEDIEV phagoStatusCroupCroops 1 = 0x21 AND			
ΖΖ.	vERIFY phaseStatusGroupGreens. I = 0x21 AND phaseStatusGroupGreens. 2 = 0x00 AND	Pass/Fall		
	phaseStatusGroupVabCalle 1 AND 0x40- 0x40			
	Note: Ensure that Phase 7 registers a Vehicle Call during 1+6 Green.			
23.	Set HITL Detector Input 18 = On			
24.	DELAY .2 Seconds			
Detector 18 extend	Is Phase 7 until Phase 2 call exists and Detector 18 gaps and Phase	e 2 call		
continues				
1.	GET vehicleDetectorExtend.18 = [currentExtendValue]			
2.	SET vehicleDetectorExtend.18 = 40			
	<i>Note:</i> Set Detector 18 extend time = 4 seconds so that actuations < 4			
	seconds apart keep the phase extending.			
3.	GET ringStatus.1, ringStatus.2			
4.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03			
5.	DELAY 1 Second			
6.	GET ringStatus.1, ringStatus.2			
7.	WEND			
	Note: Loop until controller rests somewhere.			
8.	Set HITL Detector Input 3 = On			
9.	DELAY .2 Seconds			
10.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2			
11.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND			
	phaseStatusGroupPhaseOns.2 ≠ 0x00			
12.	DELAY 1 Second			
13.	GET phaseStatusGroupPhaseOns.1,			
	phaseStatusGroupPhaseOns.2			
14.	WEND			
	Note: Wait until controller reaches 3+5.			
15.	Set HILL Detector Input 3 = Off			
16.	DELAY .2 Seconds			
17.	Set HILL Detector input 18 = On (10 call and then extend Phase 7)			
Ϊδ.	DELAY .2 Seconds			
19.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2			
20.	wHiLE pnaseStatusGroupPnaseOns.1 ≠ 0x41 AND phaseStatusGroupPhaseOns.2 ≠ 0x00			
21	DEL ΔY 1 Second			
<u> </u>	CET nhaseStatusCrounDhaseOne 1			
۲۲.	nhaseStatusGrounPhaseOne 2			
23				
20.				
	Note: Wait until controller reaches 1+7.			

24.	Set HITL Detector Input 2 = On (Detector 18 will extend Phase 4					
	when there is an opposing call)					
25.	DELAY .2 Seconds					
	Note: This should get Phase 7 to start timing extensions and sets up					
	conditional logic.					
26.	GET ringStatus.2					
27.	WHILE ringStatus.2 AND 0x07 ≠ 0x01 (xxxxx001 = extension)					
28	DELAY 1 Second					
29	GET ringStatus 2					
20. 30	WEND					
00.	WEND					
	Note: Wait for indication that extensions are timing					
	Ring 1 = $2.3 \ A \ 0.11 \ 12 \ 8.1$					
	$Ping 2 = 15 \ 16 \ 5 \ 6 \ 7 \ 8 \ 8 \ 13$					
21	$\operatorname{Ring} 2 = 15, 10, 5, 0, 7, 0, & 15$					
<u> </u>	GET pridsestatusGroupPridseOris. T					
32.		Pass/Fall				
	AND 0x40= 0x40					
	Note: Verify that the extensions are on Phase 7.					
33.	Set HILL Detector Input 18 = Off					
34.	DELAY 3 Seconds					
35.	Set HITL Detector Input 18 = On					
	DELAY .2 Seconds					
37.	Set HITL Detector Input 18 = Off					
38.	DELAY 2.8 Seconds					
39.	GET ringStatus.2					
40.	VERIFY that RESPONSE VALUE ringStatus.2 AND 0x07 = 0x01	Pass/Fail				
	(xxxxx001 = extension)					
	Note: Ring 1 = 2, 3, 4, 9, 11, 12, & 1					
	Ring $2 = 15, 16, 5, 6, 7, 8, \& 13$					
41	GET phaseStatusGroupPhaseOns 1					
42	VERIEV that RESPONSE VALUE phaseStatusGroupPhaseOns 1	Pass/Fail				
<i>τ</i> <u></u> .	$\Delta ND 0 x 40 = 0 x 40$	1 433/1 41				
	Note: Verify that extensions are still timing on Phase 7 because call					
	was entered $< A$ seconds later					
13	Sot HITL Detector Ipput 18 - On					
4J.	DELAX 2 Secondo					
44. 45	DELAT .2 Seconds					
45.	Set HITL Detector input 18 = OII					
46.	DELAY 2.8 Seconds					
47.	GET ringStatus.2					
48.	VERIFY that RESPONSE VALUE ringStatus.2 AND 0x07 = 0x01	Pass/Fail				
	(xxxxx001 = extension)					
	<i>Note:</i> Ring 1 = 2, 3, 4, 9, 11, 12, & 1					
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13					
49.	GET phaseStatusGroupPhaseOns.1					
50.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail				
	AND 0x40 = 0x40					
	Note: Verify that extensions are still timing on Phase 7 because call					
	was entered < 4 seconds later.					
51.	Set HITL Detector Input 18 = On					

52.	DELAY .2 Seconds				
53.	Set HITL Detector Input 18 = Off				
54.	DELAY 4.8 Seconds				
_					
	Note: Since the time between actuations is now 5 seconds, the timer				
	will gap and therefore disable Detector 18 from putting in any further				
	extensions.				
55.	Set HITL Detector Input 18 = On				
56.	DELAY 2 Seconds				
57	ET ringStatus.2				
58	VERIEV that RESPONSE VALUE ringStatus 2 AND $0x07 \neq 0x01$	Pass/Fail			
00.	(xxxxx001 = extension)	1 000/1 01			
	Note: Verify that Phase 7 is no longer timing extensions				
	Ring $1 = 2$ 3 4 9 11 12 & 1				
	Ring $2 = 15$ 16 5 6 7 8 & 13				
59	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupVehCalls 1				
60	WHILE ( $nhaseStatusGroupPhaseOns 1 AND 0x40 = 0x40$ ) AND				
00.	(phaseStatusGroupVehCalls 1 AND 0x40 $\neq$ 0x40)				
61	DELAY 1 Second				
62	GET nhaseStatusGrounPhaseOns 1				
02.	nhaseStatusGrounVehCalls 1				
63	WEND				
64	VERIEV (phaseStatusGroupPhaseOps 1 AND 0v40 = 0v40) AND	Pass/Fail			
04.	(nhaseStatusCroup)/ehCalls 1 AND 0x40 = 0x40) AND	1 833/1 81			
	(p) as contraction of $p$ chooses. (p) as $p = 0, +0$				
	Note: Ensure that Phase 7 registers a Vehicle Call				
65	Set HITL Detector Input 2 = $\Omega$ ff and 18 = $\Omega$ ff				
66	DELAY 2 Seconds				
	POST-CONDITION The Detector 18 vehicleDetectorExtend is still				
	set to 4 seconds				
Detector 18 extend	Is Phase 7 until Phase 2 call exists and Detector 18 gaps but not if l	Phase 2 call			
disappears					
	PRE-CONDITION The Detector 18 vehicleDetectorExtend is still set				
	to 4 seconds				
1.	GET ringStatus.1. ringStatus.2				
2.	WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03				
3	DELAY 1 Second				
4	GET ringStatus 1 ringStatus 2				
5	WEND				
0.					
	Note: Loop until controller rests somewhere				
6	Set HITL Detector Input 4 = On				
7	DELAY 2 Seconds				
8	GET phaseStatusGroupPhaseOns 1_phaseStatusGroupPhaseOns 2				
9	WHILE phaseStatusGroupPhaseOns $1 \neq 0x18$ AND				
<b>U</b> .	phaseStatusGroupPhaseOns.2 ≠ 0x00				
10.	DELAY 1 Second				
11	GET phaseStatusGroupPhaseOns 1				
	phaseStatusGroupPhaseOns 2				
12	WEND				
· <b>∠</b> .	TERE				
	Note: Wait until controller reaches 4+5.				
1.3	Set HITI Detector Input 4 = Off				

14.	DELAY .2 Seconds					
15.	Set HITL Detector Input 18 = On (To call and then extend Phase 7)					
16.	DELAY .2 Seconds					
17.	GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2				
18	WHILE phaseStatusGroupPhaseOns $1 \neq 0x41$ AND					
10.	nhaseStatusGroupPhaseOns $2 \pm 0x00$					
10	DELAV 1 Second					
20	CET nhaseStatusCrounDhaseOne 1					
20.	GET pridseSidiusGroupPhaseOns.1,					
01						
۷۱.	WEND					
	Nata Mait until controller reaches 117					
	Avoie, widit until controller reaches 1+7.					
<u> </u>	Set HITL Detector input 2 = On					
23.	DELAY .2 Seconds					
	Note: Detector 18 will extend Phase 7 to start timing extensions and					
	sets up conditional logic.					
24.	GET ringStatus.1					
25.	WHILE ringStatus.2 AND 0x07 ≠ 0x01 (xxxxx001 = extension)					
26.	DELAY .1 Second					
27.	GET ringStatus.2					
28.	WEND					
	Note: Wait for indication that extensions are timing.					
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1					
	Ring 2 = 15, 16, 5, 6, 7, 8, & 13					
29.	GET phaseStatusGroupPhaseOns.1					
30.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail				
	AND 0x40 = 0x40					
	Note: Verify that the extensions are timing on Phase 7.					
31.	Set HITL Detector Input 18 = Off					
32.	DELAY 3 Seconds					
33.	Set HITL Detector Input 18 = On					
34.	DELAY 2 Seconds					
35	Set HITL Detector Input 18 = Off					
36	DELAY 2.8 Seconds					
37	GET ringStatus 2					
38	VEDIEV that $PESPONSE VALUE ringStatus 2 AND 0x07 - 0x01$	Dass/Eail				
50.	v = 1  (Intermediate of the second secon	r ass/r all				
	Note: Wait for indication that extensions are timing					
	<i>Note:</i> Wait for indication that extensions are timing.					
	<i>Note:</i> Wait for indication that extensions are timing. Ring $1 = 2, 3, 4, 9, 11, 12, \& 1$ Ring $2 = 15, 16, 5, 6, 7, 8, \& 13$					
30	<i>Note:</i> Wait for indication that extensions are timing. Ring 1 = 2, 3, 4, 9, 11, 12, & 1 Ring 2 = 15, 16, 5, 6, 7, 8, & 13					
39.	Note: Wait for indication that extensions are timing. Ring 1 = 2, 3, 4, 9, 11, 12, & 1 Ring 2 = 15, 16, 5, 6, 7, 8, & 13 GET phaseStatusGroupPhaseOns.1	Decc/Ecil				
<u>39.</u> 40.	Note: Wait for indication that extensions are timing. Ring 1 = 2, 3, 4, 9, 11, 12, & 1 Ring 2 = 15, 16, 5, 6, 7, 8, & 13 GET phaseStatusGroupPhaseOns.1 VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 AND 0x40 = 0x40	Pass/Fail				
<u>39.</u> 40.	Note: Wait for indication that extensions are timing. Ring 1 = 2, 3, 4, 9, 11, 12, & 1 Ring 2 = 15, 16, 5, 6, 7, 8, & 13 GET phaseStatusGroupPhaseOns.1 VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 AND 0x40 = 0x40	Pass/Fail				
<u>39</u> . 40.	Note: Wait for indication that extensions are timing. Ring 1 = 2, 3, 4, 9, 11, 12, & 1 Ring 2 = 15, 16, 5, 6, 7, 8, & 13 GET phaseStatusGroupPhaseOns.1 VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 AND 0x40 = 0x40	Pass/Fail				
<u> </u>	Note: Wait for indication that extensions are timing. Ring 1 = 2, 3, 4, 9, 11, 12, & 1 Ring 2 = 15, 16, 5, 6, 7, 8, & 13 GET phaseStatusGroupPhaseOns.1 VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 AND 0x40 = 0x40 Note: Verify that extensions are still timing is on Phase 7.	Pass/Fail				
<u>39.</u> 40. <u>41.</u>	Note: Wait for indication that extensions are timing. Ring 1 = 2, 3, 4, 9, 11, 12, & 1 Ring 2 = 15, 16, 5, 6, 7, 8, & 13 GET phaseStatusGroupPhaseOns.1 VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 AND 0x40 = 0x40 Note: Verify that extensions are still timing is on Phase 7. Set HITL Detector Input 18 = On	Pass/Fail				
39. 40. 41. 42.	Note: Wait for indication that extensions are timing. Ring 1 = 2, 3, 4, 9, 11, 12, & 1 Ring 2 = 15, 16, 5, 6, 7, 8, & 13 GET phaseStatusGroupPhaseOns.1 VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 AND 0x40 = 0x40 Note: Verify that extensions are still timing is on Phase 7. Set HITL Detector Input 18 = On DELAY .2 Seconds	Pass/Fail				
39. 40. 41. 42. 43.	Note: Wait for indication that extensions are timing. Ring 1 = 2, 3, 4, 9, 11, 12, & 1 Ring 2 = 15, 16, 5, 6, 7, 8, & 13 GET phaseStatusGroupPhaseOns.1 VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 AND 0x40 = 0x40 Note: Verify that extensions are still timing is on Phase 7. Set HITL Detector Input 18 = On DELAY .2 Seconds Set HITL Detector Input 18 = Off	Pass/Fail				
39. 40. 41. 42. 43. 44.	Note: Wait for indication that extensions are timing. Ring 1 = 2, 3, 4, 9, 11, 12, & 1 Ring 2 = 15, 16, 5, 6, 7, 8, & 13 GET phaseStatusGroupPhaseOns.1 VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1 AND 0x40 = 0x40 Note: Verify that extensions are still timing is on Phase 7. Set HITL Detector Input 18 = On DELAY .2 Seconds Set HITL Detector Input 18 = Off DELAY 2.8 Seconds	Pass/Fail				

46.	<ol> <li>VERIFY that RESPONSE VALUE ringStatus.2 AND 0x07 = 0x01 (xxxxx001 = extension)</li> </ol>			
	Note: Wait for indication that extensions are timing.			
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1			
	Ring $2 = 15, 16, 5, 6, 7, 8, \& 13$			
47.	GET phaseStatusGroupPhaseOns.1			
48.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail		
	AND 0x40 = 0x40			
	Note: Verify that extensions are still timing on Phase 7.			
49.	Set HITL Detector Input 18 = On			
50.	ELAY .2 Seconds			
51.	Set HITL Detector Input 18 = Off, 7 = On, 2 = Off and 3 = On			
	Note: Stop extending Phase 7 via Detector 18, turn on Detector 7 to			
	keep Phase 7 extending, turn off Detector 2 opposing demand as			
	part of the logic, and turn on Detector 3 to place opposing demand.			
	Detector 7 will keep extending Phase 7 irrespective of Detector 18,			
	no call on Phase 2 will reset Detector 18 gap function, and the call			
-	on Phase 3 will enable extensions to continue to time.			
52.	DELAY 4.8 Seconds			
	Note: This would have the effect of allowing the extend timer to gap			
	and therefore disable Detector 18, but since Phase 2 no longer has a			
	call, another activation of Detector 18 will continue to extend Phase			
<b>5</b> 2	7. Set HITL Detector Input 19 = On and 7 = Off			
<u> </u>	DELAX 2 Seconds			
55	Sat HITL Detector Input 18 - Off			
55.	DELAV 2 Seconds			
57	CET ringStatus 2			
58	VERIEV that RESPONSE VALUE ringStatus 2 AND $0x07 = 0x01$	Pass/Fail		
50.	(xxxxx001 = extension)	r ass/r all		
	Note: Wait for indication that extensions are timing.			
	Ring 1 = 2, 3, 4, 9, 11, 12, & 1			
	Ring $2 = 15, 16, 5, 6, 7, 8, \& 13$			
59.	GET phaseStatusGroupPhaseOns.1			
60.	VERIFY that RESPONSE VALUE phaseStatusGroupPhaseOns.1	Pass/Fail		
	AND 0x40 = 0x40			
	Note: Verify that extensions are still timing on Phase 7 because even			
	though call was > 4 seconds later, the absence of call on phase 2			
	allowed phase 7 to continue to time extensions.			
61.	Set HITL Detector Input 3 = Off			
62.	DELAY .2 Seconds			
63.	SET vehicleDetectorExtend.18 = [currentExtendValue]			
	Note: Return Detector 18 extend time back to their original value.			
Detector 18 extend	Is Phase 15 when Phase 15 Green			
1.	GET ringStatus.1, ringStatus.2			
2.	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03			
3.	DELAY 1 Second			
4.	GET ringStatus.1, ringStatus.2			

5.	WEND					
	Note: Loop until controller rests somewhere.					
6.	Set HITL Detector Input 7 = On					
7.	DELAY .2 Seconds					
8.	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2					
9.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x41 AND					
	phaseStatusGroupPhaseOns.2 ≠ 0x00					
10.	DELAY 1 Second					
11.	GET phaseStatusGroupPhaseOns.1.					
	phaseStatusGroupPhaseOns.2					
12.	WEND					
	Note: Wait until controller reaches 1+7.					
13.	Set HITL Detector Input 7 = Off					
14.	DELAY .2 Seconds					
15.	Set HITL Detector Input 2 = On					
16.	DELAY 2 Seconds					
17	GET phaseStatusGroupPhaseOns 1 phaseStatusGroupPhaseOns 2					
18	WHILE phaseStatusGroupPhaseOns $1 \neq 0x02$ AND					
10.	phaseStatusGroupPhaseOns 2 ≠ 0x40					
19.	DELAY 1 Second					
20	GET phaseStatusGroupPhaseOns 1					
_0.	phaseStatusGroupPhaseOns 2					
21	WEND					
	Note: Wait until controller reaches 2+15.					
22	Set HITL Detector Input 2 = Off					
23	DELAY 2 Seconds					
24	Set HITL Detector Input 18 = On					
25	DELAY .2 Seconds					
26	GET ringStatus.2					
27	WHILE ringStatus 2 AND $0x10 \neq 0x10$ (xxx1xxxx = maxout)					
27.	DELAY 1 Second					
20.	GET ringStatus 2					
30						
50.	WEND					
	Note: Wait for Max Out Indication					
	Ring 1 = 2 3 4 9 11 12 & 1					
	$Ring 2 = 15 \ 16 \ 5 \ 6 \ 7 \ 8 \ 8 \ 13$					
31	GET nhaseStatusGrounPhaseOns 2					
32	VEDIEV that DESDONSE VALUE phaseStatusCroupDhaseOne 2					
52.	$\Delta ND 0 y 40 = 0 y 40$	1 833/1 81				
	Note: Ensure that Max Out occurred on Phase 15					
33	Set HITL Detector Input 18 = Off					
3 <u>4</u>	DELAV 2 Seconds					
Teardown	DELAT 2 Geoonds					
Teardown	REPEORM Detector Teardown TC020 if not proceeding to another					
	detector operation test case					
-						
Tested Bv:	Date					
,						

<ul> <li>v1.00 05/09/06 Initial Draft – RDR</li> <li>v1.01 07/03/06 Updated logic – RDR</li> <li>v1.02 07/05/06 Updated notes – RDR</li> <li>v1.03 08/17/06 Implemented script and proofed – JJ</li> </ul>

## Detector Operations Setup

Test Case:	Title:	Detector Operations Setup				
TC019	Description:	This procedure performs general setup of controller parameters				
		to facilitate testing and provide consistent operation	n.			
	Constants:					
	Variables:	[currentMinGrn.Phase]				
		[currentPassage.Phase]				
		[currentMax1.Phase]				
	Pass/Fail	The DUT shall pass every verification step include	d within the			
	Criteria:	Test Case in order to pass the Test Case.				
Test Step	Test Procedure		Results			
Number						
Setup						
1.	CONFIGURE the c	controller for:				
	1. 4-phase diamoi	nd operation with Phase 3 and 7 in sequence				
	2. Connection to 1	S 2 Tester Box (BIU's enabled)				
	3. Vehicle and Pe	destrian Recalls are all off				
	4. Default data loa	aded				
	5. Call non-locking	g memory is set so that calls are not remembered				
	if a call is removed					
	<ol> <li>6. Yellow clearances are set to at least 3.5 seconds</li> <li>6. Red clearances are set to at least 1.5 seconds</li> </ol>					
	Mater The action fo					
	Note: The setup to	r different manufacturer's controller's will be				
	different. Referend	ce should be made to a document containing such				
	Information.	n Notoo:				
		II NOLES.				
	Unit Data – Sta	$f(u) \propto m(s) = A(t) Sequence = 10$				
	ond Malfunction	LIDALA – ENADIE I &F I-4, DEI I-4,				
	Bhase Date	riticlization & N.A. Boononee Dheese 4 and 7 -				
	Change "Data – II	TO "Inpotino"				
	Change Dark					
	Naztec Configurati	on Notes:				
	Econolite Notes:					
2	FOR Phase = 1 T(	0.16				
3.	GET phaseMin	imumGreen Phase, phasePassage, Phase, and				
5.	phaseMaximun	11.Phase				
4.	RECORD RES	PONSE VALUE in [currentMinGrn.Phase].				
	[currentPassag	e.Phase] and [currentMax1.Phase]				
	Letter and a second					
	Note: These va	lues will be restored at the end of the test case.				

5.	SET phaseMinimumGreen.Phase = 5, phasePassage.Phase = 0, and phaseMaximum1.Phase = 25 <i>Note:</i> min = 5 second, passage = 4 seconds, and max = 25 seconds.			
6.	NEXT Phase			
Test Case Results				
Tested By:	Date Tested	Pass/Fail		
Test Case Notes:	<notes></notes>			
Version History:	<ul> <li>v1.00 05/05/06 Initial Draft – RDR</li> <li>v1.01 06/27/06 Updated test values – RDR</li> <li>v1.02 07/05/06 Updated notes – RDR</li> <li>v1.03 07/27/06 Implemented script and proofed – JJ</li> </ul>			

Detector Operations Teardown

Test Case:	Title:	Detector Ope	erations T	Feardown	tors aftar
10020	Description.	executing Detector Operation Setup – TC001			
	Constants:				
	Variables:				
	Pass/Fail	The DUT sha	ll pass ev	ery verification step include	d within the
	Criteria:	Test Case in	order to p	ass the Test Case.	
Test Step	Test Procedure				Results
Number					
Teardown					
1.	FOR Phase = 1 TO 16				
2.	SET phaseMinimumGreen.Phase = [currentMinGrn.Phase], Pass/Fail			Pass/Fail	
	phasePassage.Phase = [currentPassage.Phase],				
	phaseMaximum1.Phase = [currentMax1.Phase]				
3.	NEXT Phase				
		Test Case	Results		
Tested By:			Date		Pass/Fail
			lested		
Test Case Notes:	<notes></notes>				
Version History:	v1.00 05/08/06 Initial Draft – RDR				
	v1.01 07/27/06 Implemented script and proofed – JJ				

Detector Delay

Test Case: TC021	<i>Title</i> : Description:	<b>Detector Delay</b> Verifies that, when programmed, Detectors 2, 3, 4, 6, 7, and 8 delay entering a call for the parent phase when the parent phase is red.
	Constants: Variables: Pass/Fail Criteria:	currentDetectorDelay.2 (.4, 6, 7, and 8) The DUT shall pass every verification step included within the Test Case in order to pass the Test Case.

Number	Number         FOR DetectorNumber = 2, 3, 4, 6, 7, and 8           GET vehicleDetectorDelay.DetectorNumber         RECORD RESPONSE VALUE in [currentDetectorDelay.DetectorNumber]           More: These values will be restored at the end of the test case.         SET vehicleDetectorDelay.DetectorNumber = 2           Pass/Fail         NEXT Phase           Detector 2 Delay         WHILE ringStatus 1, ringStatus 2           WHILE ringStatus 1, ringStatus 2         WWILE ringStatus 1, ringStatus 2           WEND         WEND           Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall 1 = 0x20           DELAY 1 Second         Set phaseControlGroupVehCall 1 = 0x00           GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2         WHILE phaseStatusGroupPhaseOns 1 + 0x21 AND           phaseStatusGroupPhaseOns 1 + 0x21 AND         phaseStatusGroupPhaseOns 2 + 0x00           DELAY 1 Second         GET phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2           WEND         Note: Weit until controller reaches 1+6.           Set phaseStatusGroupPhaseOns 2         0X00           DELAY 1 Second         DELAY 1 Second           GET phaseStatusGroupPhaseOns 1, phaseStatusGroupVehCalls.1         Pass/Fail           Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On           DELAY 1.9 Seconds         DELAY 1.9 Seconds	Test Step	Test Procedure	Results
Setup           FOR DetectorNumber = 2, 3, 4, 6, 7, and 8	Setup           FOR DetectorNumber = 2, 3, 4, 6, 7, and 8           GET vehicleDetectorDelay, DetectorNumber           RECORD RESPONSE VALUE in [currentDetectorDelay, DetectorNumber]           Note: These values will be restored at the end of the test case.           SET vehicleDetectorDelay, DetectorNumber = 2           Pass/Fail           NEXT Phase           Detector 2 Delay           WHILE ringStatus.1, ringStatus.2           WHILE ringStatus.1, ringStatus.2           WEND           GET ringStatus.1, ringStatus.2           WEND           OELAY 1 Second           GET ringStatus.1, ringStatus.2           WEND           Note: Loop until controller rests in green somewhere.           Set phaseControlGroupVehCall.1 = 0x20           DELAY 2 Seconds           Set phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2           WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2           WHILE phaseStatusGroupPhaseOns.2 = 0X00           DELAY 1 Second           GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseStatusGroupPhaseOns.2           WEND           Note: Wait until controller reaches 1+6.           Set HITL Detector Input 2 = 0n           DELAY 1.9 Seconds           GET phaseStatusGroupVehCalls.1	Number		
FOR DetectorNumber = 2, 3, 4, 6, 7, and 8	FOR DetectorNumber = 2, 3, 4, 6, 7, and 8         GET vehicleDetectorDelay.DetectorNumber         RECORD RESPONSE VALUE in [currentDetectorDelay.DetectorNumber]         Note: These values will be restored at the end of the test case. SET vehicleDetectorDelay.DetectorNumber = 2         Pass/Fail         NEXT Phase         Detector 2 Delay         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1, *003 AND ringStatus.2 * 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 1 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 * 0x21 AND         phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1         phaseStatusGroupPhaseOns.2         WEND         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.	Setup		
FOR DetectorNumber 2, 3, 4, 6, 7, and 8         GET vehicleDetectorDelay.DetectorNumber         RECORD RESPONSE VALUE in [currentDetectorDelay.DetectorNumber]         Note: These values will be restored at the end of the test case.         SET vehicleDetectorDelay.DetectorNumber = 2         Pass/Fail         NEXT Phase         Detector 2 Delay         Control Group Vehicle DetectorNumber = 2         WHILE ingStatus.1, ringStatus.2         WHILE ingStatus.1, ringStatus.2         WHILE ingStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE         OET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 # 0x20         DELAY 1 Seconds         GET phaseStatusGroupPhaseOs.1         VERIFY that RESPONSE VAL	FOR DetectorNumber = 2, 3, 4, 6, 7, and 8         GET vehicleDetectorDelay.DetectorNumber         RECORD RESPONSE VALUE in         [currentDetectorDelay.DetectorNumber]         Note: These values will be restored at the end of the test case.         SET vehicleDetectorDelay.DetectorNumber = 2         Pass/Fail         NEXT Phase         Detector 2 Delay         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET pringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 2. Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET ringStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, optaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, optaseStatusGroupPhaseOns.2         WEND         Note: Weit until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 1.9 Seconds         GET phaseStatusGrou			
GET vehicleDetectorDelay.DetectorNumber         RECORD RESPONSE VALUE in [currentDetectorDelay.DetectorNumber]         Note: These values will be restored at the end of the test case.         SET vehicleDetectorDelay.DetectorNumber = 2         Pass/Fail         NEXT Phase         Detector 2 Delay         GET ringStatus.1, # 0x03 AND ringStatus.2 # 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 2. Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 + 0x21 AND         phaseStatusGroupPhaseOns.1 + 0x21 AND         phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 + 0x21 AND         phaseStatusGroupPhaseOns.1         Mote: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Wait until controller reaches 1+6.         Set HITL Detect	GET vehicleDetectorDelay.DetectorNumber         RECORD RESPONSE VALUE in [currentDetectorDelay.DetectorNumber]         Note: These values will be restored at the end of the test case.         SET vehicleDetectorDelay.DetectorNumber = 2         Pass/Fail         NEXT Phase         Detector 2 Delay         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 2 Seconds         Set phaseControlGroupVehCall.1 = 0x20         DELAY 2 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 # 0x21 AND         phaseStatusGroupPhaseOns.2 # 0X00         DELAY 1.2 Second         GET phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseSt		FOR DetectorNumber = 2, 3, 4, 6, 7, and 8	
RECORD RESPONSE VALUE in [currentDetectorDelay.DetectorNumber]       Note: These values will be restored at the end of the test case.         SET vehicleDetectorDelay.DetectorNumber = 2       Pass/Fail         NEXT Phase       Pass/Fail         Detector 2 Delay       Image: Comparison of the test case.         WHILE ringStatus.1, ringStatus.2       Image: Comparison of the test case.         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03       Image: Comparison of the test case.         WHILE ringStatus.1, ringStatus.2       Image: Comparison of test case.         WEND       CET ringStatus.1, ringStatus.2       Image: Comparison of test case.         Note: Loop until controller rests in green somewhere.       Image: Comparison of test case.       Image: Comparison of test case.         Set phaseControlGroupVehCall.1 = 0x20       Image: Comparison of test case.       Image: Comparison of test case.       Image: Comparison of test case.         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       Image: Comparison of test case.       Image: Comparison of test case.       Image: Comparison of test case.         WHILE phaseStatusGroupPhaseOns.2       Image: Comparison of test case.       Image: Comp	RECORD RESPONSE VALUE in [currentDetectorDelay.DetectorNumber]         Note: These values will be restored at the end of the test case.         SET vehicleDetectorDelay.DetectorNumber = 2         Pass/Fail         NEXT Phase         Detector 2 Delay         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ± 0x03 AND ringStatus.2 ± 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, 0x21 AND         phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, 0x21 AND         phaseStatusGroupPhaseOns.2         WHD         VEND         Set HITL Detector Input 2 = 0n         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that detector call is not active yet.         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call		GET vehicleDetectorDelay.DetectorNumber	
Image: Interpret (current/Delay.Detector/Number)       Image: Image	[currentDetectorDelay.DetectorNumber]         Note: These values will be restored at the end of the test case.         SET vehicleDetectorDelay.DetectorNumber = 2         Pass/Fail         NEXT Phase         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 + 0x03 AND ringStatus.2 ≠ 0x03         DEtector 2 Delay         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 + 0x21 AND         phaseStatusGroupPhaseOns.2 + 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VX00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VARIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         D		RECORD RESPONSE VALUE in	
Note: These values will be restored at the end of the test case.       SET vehicleDetectorDelay.DetectorNumber = 2       Pass/Fail         NEXT Phase       Pass/Fail         Detector 2 Delay       Image: Comparison of the test case.         WHILE ringStatus.1, ringStatus.2       Image: Comparison of the test case.         WHILE ringStatus.1, ringStatus.2       Image: Comparison of the test case.         WEND       Image: Comparison of the test case.         Note: Loop until controller rests in green somewhere.       Image: Comparison of the test case.         Set phaseControlGroupVehCall.1 = 0x20       Image: Comparison of test case.         DELAY.2 Seconds       Image: Comparison of test case.         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       Image: Comparison of test case.         WHILE phaseStatusGroupPhaseOns.2 ≠ 0X00       Image: Comparison of test case.         DELAY 1 Second       Image: Comparison of test case.         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2.       Image: Comparison of test case.         WEND       Note: Wait until controller reaches 1+6.       Image: Comparison of test case.         DELAY 1.9 Seconds       Image: Comparison of test case.       Image: Comparison of test case.         DELAY 1.9 Seconds       Image: Comparison of test case.       Image: Comparison of test case.         Mote: Verify that detector call is not active y	Note: These values will be restored at the end of the test case.       SET vehicleDetectorDelay DetectorNumber = 2       Pass/Fail         NEXT Phase       NEXT Phase         Detector 2 Delay       GET ringStatus.1, ringStatus.2       NUMILE ringStatus.1, ringStatus.2         WHILE ringStatus.1, ringStatus.2       WHILE ringStatus.1, ringStatus.2       WEND         Note: Loop until controller rests in green somewhere.       Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20       DELAY .2 Seconds       Set phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 + 0x21 AND       phaseStatusGroupPhaseOns.1 + 0x21 AND       phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 // WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 // WEND       Note: Wait until controller reaches 1+6.       Set HITL Detector Input 2 = On         DELAY 1.9 Seconds       GET phaseStatusGroupVehCalls.1       Pass/Fail         VCRNIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00       Note: Verify that detector call is not active yet.       Pass/Fail         DELAY 2 Seconds       GET phaseStatusGroupVehCalls.1       Pass/Fail         VCRNIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02       Note: Verify that detector call is now active.       Pass/Fail		[currentDetectorDelay.DetectorNumber]	
Note:       These values will be restored at the end of the test case.       Pass/Fail         NEXT Phase       Pass/Fail         NEXT Phase       Pass/Fail         Detector 2 Delay       GET ringStatus.1, ringStatus.2       0x03         DELAY 1 Second       GET ringStatus.1, ringStatus.2       0x03         WHILE ringStatus.1, ringStatus.2       WEND       0         Note:       Loop until controller rests in green somewhere.       0         Set phaseControlGroupVehCall.1 = 0x20       0       0         DELAY 1.2 Seconds       0       0         Set phaseControlGroupVehCall.1 = 0x00       0       0         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       0       0         WHILE phaseStatusGroupPhaseOns.1 + 0x21 AND       0       0         phaseStatusGroupPhaseOns.1 + 0x21 AND       0       0         DELAY 1 Second       0       0       0         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       0       0       0         WEND       0       0       0       0       0       0         DELAY 1 Second       0       0       0       0       0       0       0       0       0       0       0       0       0       0	Note:       Ihese values will be restored at the end of the test case.       Pass/Fail         NEXT Phase       Pass/Fail         Detector 2 Delay       GET ringStatus.1, ringStatus.2       0x03         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03       DELAY 1 Second       GET ringStatus.1, ringStatus.2         WEND       WEND       Mote: Loop until controller rests in green somewhere.       Set phaseControlGroupVehCall.1 = 0x20         DELAY.2 Seconds       Set phaseControlGroupPvehCall.1 = 0x20       GET ringStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       WHILE phaseStatusGroupPhaseOns.1 + 0x21 AND         phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       WEND         Note: WeND       Note: Verify haseOns.1, phaseStatusGroupPhaseOns.2         WEND       WEND       Work: WenD         Note: Wait until controller reaches 1+6.       WEND         Note: Wait until controller reaches 1+6.       Set HITL Detector Input 2 = On         DELAY 1.9 Seconds       GET phaseStatusGroupVehCalls.1         GET phaseStatusGroupVehCalls.1       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00       Note: Verify that detector call is not active yet.         DELAY 2. Seconds       GET phaseStatusGroupVehCalls.1       Pass/Fail         VERIFY that RESPONSE VALU		····	
SE 1 vehicle/Detector/Delay.Detector/Number = 2       Pass/Fail         NEXT Phase       Pass/Fail         Detector 2 Delay       GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	SE I vehicleDetectorDelay.DetectorNumber = 2       Pass/Fail         NEXT Phase       Pass/Fail         Detector 2 Delay       GET ringStatus.1, ringStatus.2         WHILE ringStatus.1, ringStatus.2       \$\pmathcal{V}\$ 0x03 AND ringStatus.2 \$\pmathcal{V}\$ 0x03         DELAY 1 Second       GET ringStatus.1, ringStatus.2         WEND       Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20       DELAY .2 Seconds         Set phaseControlGroupVehCall.1 = 0x00       GET ringStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND       Mote: Wait until controller reaches 1+6.       Pass/Fail         Note: Wait until controller reaches 1+6.       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         Note: Verify that detector call is not active yet.       Pass/Fail         DELAY .2 Seconds       GET phaseStatusGroupVehCalls.1       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         VC2       Note: Verify that detector call is now active.       Pass/Fail         VO2       Note: Verify		Note: These values will be restored at the end of the test case.	
NEXT Phase	NEX1 Phase         Detector 2 Delay         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY.2 Seconds         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY.2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2. Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is now active.         DELAY 2. Seconds         GET phaseStatusGroupVehC		SET vehicleDetectorDelay.DetectorNumber = 2	Pass/Fail
Detector 2 Delay       GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 1.2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET ringStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseSta	Detector 2 Delay         GET ringStatus.1, ringStatus.2         WHILE ringStatus.1, ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 1.2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2. Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is now active yet.         DELAY 2. Seconds         GET phaseStatusGroupVehCalls.1         V		NEXTPhase	
GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY.2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, ± 0x21 AND         phaseStatusGroupPhaseOns.1, ± 0x21 AND         phaseStatusGroupPhaseOns.2 ± 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds       E         GET phaseStatusGroupVehCalls.1       Pass/Fail         0x00       Note: Verify that detector call is not active yet.	GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 2 Seconds         Set phaseControlGroupVehCall.1 = 0x20         DELAY 2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VCRIFY that RES			
WHILE ringStatus. 2       0x03 AND ringStatus. 2 ≠ 0x03         DELAY 1 Second	GET IntgStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         WHILE ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 1 Seconds         Set phaseControlGroupVehCall.1 = 0x20         DELAY 2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 1.2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 1.2 Seconds       EGT phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00       Pass/Fail         Note: Verify that detector call is now active.       Pass/Fail	Detector 2 Delay	OFT size Otatus 4, size Otatus 0	
WHILE InfgStatus.1 # 0x03 AND InfgStatus.2 # 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 2 Seconds         GET phaseControlGroupVehCall.1 = 0x00         GET phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 # 0x21 AND         phaseStatusGroupPhaseOns.2 # 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1	WHILE InfgStatus 1 ≠ 0x00 AND InfgStatus 2 ≠ 0x03         DELAY 1 Second         GET ringStatus 1, ringStatus 2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall 1 = 0x20         DELAY 2 Seconds         Set phaseStatusGroupPhaseOns 1, phaseStatusGroupPhaseOns 2         WHILE phaseStatusGroupPhaseOns 1 ≠ 0x21 AND         phaseStatusGroupPhaseOns 2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns 2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02         Note: Verify that detector call is now activ		GET ringStatus.1, ringStatus.2	
OELAY 1 Second       GET ringStatus.1, ringStatus.2         WEND       Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20       DELAY .2 Seconds         GET phaseStatusGroupVehCall.1 = 0x00       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       WHILE phaseStatusGroupPhaseOns.1 # 0x21 AND         phaseStatusGroupPhaseOns.1 # 0x21 AND       phaseStatusGroupPhaseOns.1 # 0x21 AND         phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 # 0x00       DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 # 0x00       DELAY 1 Second         WEND       Note: Wait until controller reaches 1+6.       Set HITL Detector Input 2 = On         DELAY 1.9 Seconds       GET phaseStatusGroupVehCalls.1       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00       Note: Verify that detector call is not active yet.       Pass/Fail         DELAY .2 Seconds       GET phaseStatusGroupVehCalls.1       Pass/Fail       VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00	DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         DELAY.2 Seconds         Set phaseControlGroupVehCall.1 = 0x20         DELAY.2 Seconds         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.1,         gET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Weit until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         VAC         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
WEND       Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20       DELAY .2 Seconds         GET phaseStatusGroupVehCall.1 = 0x00       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.1 ≠ 0x21 AND       phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second       GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND       Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On       DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1       VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.       Pass/Fail         DELAY .2 Seconds       GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00       Pass/Fail         Note: Verify that detector call is not active yet.       Pass/Fail         DELAY .2 Seconds       GET phaseStatusGroupVehCalls.1       Pass/Fail         0x02       Note: Verify that detector call is not active yet.       Pass/Fail	WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY.2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		DELAY 1 Second	
WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY 2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02         Note: Verify that detector call is not active yet.	Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY .2 Seconds         GET phaseControlGroupVehCall.1 = 0x00         GET phaseControlGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off			
Note: Loop until controller rests in green somewhere.       Image: Control Group VehCall.1 = 0x20         DELAY .2 Seconds       Image: Control Group VehCall.1 = 0x00         Set phaseControl Group VehCall.1 = 0x00       Image: Control Group VehCall.1 = 0x00         GET phaseStatusGroup PhaseOns.1, phaseStatusGroup PhaseOns.2       Image: Control Group VehCall.1 = 0x00         GET phaseStatusGroup PhaseOns.1 ≠ 0x21 AND       Image: Control Get PhaseStatusGroup PhaseOns.1 ≠ 0x21 AND         phaseStatusGroup PhaseOns.2 ≠ 0x00       Image: Control Get PhaseStatusGroup PhaseOns.1, phaseStatusGroup PhaseOns.2         DELAY 1 Second       Image: Control Get PhaseStatusGroup PhaseOns.2         WEND       VeND         Note: Wait until controller reaches 1+6.       Image: Control Get PhaseStatusGroup VehCalls.1         DELAY 1.9 Seconds       Image: Control Get PhaseStatusGroup VehCalls.1         GET phaseStatusGroup VehCalls.1       Image: Control Get PhaseStatusGroup VehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroup VehCalls.1 = 0x00       Image: Control Get PhaseStatusGroup VehCalls.1         Note: Verify that detector call is not active yet.       Image: Control Get PhaseStatusGroup VehCalls.1         DELAY 2 Seconds       Image: Control Get PhaseStatusGroup VehCalls.1       Image: Control Get PhaseStatusGroup VehCalls.1         Mote: Verify that detector call is not active yet.       Image: Control Get PhaseStatusGroup VehCalls.1       Image: Control Get PhaseStatu	Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY.2 Seconds         GET phaseControlGroupVehCall.1 = 0x00         GET phaseControlGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, wEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02         Note: Verify that detector call is now active.         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		WEND	
Note: Loop unit controler resis in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY .2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02         Note: Verify that detector call is now active	Note: Loop untit controller resis in green somewhere.         Set phaseControlGroupVehCall.1 = 0x20         DELAY.2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHIL phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		Nota: Loop until controller rests in green somewhere	
Set phaseControlGroupVehCall.1 = 0x20	Set phaseControlGroupVehCall.1 = 0x20         DELAY .2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2 / 0x00         DELAY 1 Second         GET weith the controller reaches 1+6.         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		The solution of the second of	
DELAY 2 Seconds	DeLAY 2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2. Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2. Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		Set phaseControlGroup\/ebCall 1 = 0v20	
Set phaseControlGroupVehCall.1 = 0x00         GET phaseControlGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELLAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02         Note: Verify that detector call is now active	Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02         Note: Verify that detector call is now active.         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		DELAY 2 Seconds	
GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x20         phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2,         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is now active	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2,         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         VERIFY that detector call is now active.         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		Set phaseControlGroupVebCall 1 = 0x00	
GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is now active	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY 2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is now active.         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off			
WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is not active yet.	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND         phaseStatusGroupPhaseOns.2 ≠ 0x00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		GET phaseStatusGroupPhaseOns.1. phaseStatusGroupPhaseOns.2	
phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0X00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VCR: Verify that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         Vote: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         VAD         VAD         VAD         VAD	phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is now active.         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		WHILE phaseStatusGroupPhaseOns.1 ≠ 0x21 AND	
DELAY 1 Second	DELAY 1 Second         GET phaseStatusGroupPhaseOns.1,         phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		phaseStatusGroupPhaseOns.2 ≠ 0X00	
GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       Image: Composition of the second	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2       Image: Control of the second seco		DELAY 1 Second	
phaseStatusGroupPhaseOns.2       Image: Complexity of the second se	phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         VAD         Note: Verify that detector call is now active.         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		GET phaseStatusGroupPhaseOns.1,	
WEND Note: Wait until controller reaches 1+6.	WEND       Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On		phaseStatusGroupPhaseOns.2	
Note: Wait until controller reaches 1+6.	Note: Wait until controller reaches 1+6.         Set HITL Detector Input 2 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         Vote: Verify that detector call is now active.         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		WEND	
Set HITL Detector Input 2 = On       Image: Constraint of the set of t	Set HITL Detector Input 2 = On		Note: Wait until controller reaches 1+6.	
Set HITL Detector Input 2 = On	Set HITL Detector Input 2 = On			
DELAY 1.9 Seconds	DELAY 1.9 Seconds		Set HITL Detector Input 2 = On	
DELAY 1.9 Seconds       GET phaseStatusGroupVehCalls.1         GET phaseStatusGroupVehCalls.1       VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00         Note: Verify that detector call is not active yet.       Pass/Fail         DELAY .2 Seconds       DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1       Pass/Fail         Note: Verify that detector call is now active       Pass/Fail	DELAY 1.9 Seconds       GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         0x00       Note: Verify that detector call is not active yet.         DELAY .2 Seconds       DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         Note: Verify that detector call is now active.			
GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x00         Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is now active	GET phaseStatusGroupVehCalls.1       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         Note: Verify that detector call is not active yet.       Pass/Fail         DELAY .2 Seconds       Pass/Fail         GET phaseStatusGroupVehCalls.1       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         Note: Verify that detector call is now active.       Pass/Fail         Set HITL Detector Input 2 = Off       Pass/Fail		DELAY 1.9 Seconds	
VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         0x00       Note: Verify that detector call is not active yet.         DELAY .2 Seconds       Image: Verify that RESPONSE VALUE phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         0x02       Note: Verify that detector call is now active	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         0x00       Note: Verify that detector call is not active yet.         DELAY .2 Seconds       Image: Compare StatusGroupVehCalls.1         GET phaseStatusGroupVehCalls.1       VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         V002       Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off       Image: Compare StatusCompute 2 = Off		GET phaseStatusGroupVehCalls.1	
0x00       Note: Verify that detector call is not active yet.         DELAY .2 Seconds       Image: Verify that detector call is now active         Umage: Verify that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x02       Pass/Fail	0x00       Note: Verify that detector call is not active yet.         DELAY .2 Seconds       Image: Comparison of the second secon		VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =	Pass/Fail
Note: Verify that detector call is not active yet.	Note: Verify that detector call is not active yet.		0x00	
Note: Verify that detector call is not active yet.         DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is now active	DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		Nate: Varify that datastar call is not active yet	
DELAY .2 Seconds GET phaseStatusGroupVehCalls.1 VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = Pass/Fail 0x02 Note: Verify that detector call is now active	DELAY .2 Seconds       GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         0x02       Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off       Endet		<i>Note:</i> Verify that detector call is not active yet.	
DELAY .2 Seconds         GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is now active	GET phaseStatusGroupVehCalls.1         VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =         0x02         Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		DELAX 2 Secondo	
VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = Pass/Fail 0x02	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =       Pass/Fail         0x02       Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off       Image: Comparison of the set of the		DELAT .2 Seconds	
0x02	VERIFIC that RESPONSE VALUE phaseStatusGroupVenCalls.1 =       Pass/Fall         0x02       Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off		VEDIEV that DESDONISE VALUE phaseStatusCroupVabCalls 1 -	Dace/Eail
Note: Verify that detector call is now active	Note: Verify that detector call is now active.       Set HITL Detector Input 2 = Off			Fass/Fall
Note: Verify that detector call is now active	Note: Verify that detector call is now active.         Set HITL Detector Input 2 = Off			
	Set HITL Detector Input 2 = Off		Note: Verify that detector call is now active	
	Set HITL Detector Input 2 = Off			
Set HITL Detector Input 2 = Off			Set HITL Detector Input 2 = Off	

Detector 3 Delay		
	GET ringStatus.1, ringStatus.2	
-	WHILE ringStatus.1 $\neq$ 0x03 AND ringStatus.2 $\neq$ 0x03	
	DELAY 1 Second	
	GET ringStatus.1, ringStatus.2	
	WEND	
	Note: Loop until controller rests in green somewhere	
	Set phaseControlGroupVehCall 1 = 0x40	
	DELAV 2 Seconds	
	Set phaseControlGroupVebCall 1 = 0v00	
	while phaseStatusGroupPhaseOns. 1 ≠ 0x41 AND	
	pnaseStatusGroupPnaseOns.2 ≠ 0X00	
	DELAY 1 Second	
	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
	WEND	
	Note: Wait until controller reaches 1+7.	
	Set HITL Detector Input 3 = On	
	DELAY 1.9 Seconds	
	GET phaseStatusGroupVehCalls.1	
	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =	Pass/Fail
	0x00	
	Note: Verify that detector call is not active yet.	
	DELAY .2 Seconds	
	GET phaseStatusGroupVehCalls.1	
	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =	Pass/Fail
	0x04	
	Note: Verify that detector call is now active.	
	······	
	Set HITL Detector Input 3 = Off	
Detector 4 Delay		
Deteotor + Delay	GET ringStatus 1 ringStatus 2	
	WHILE ringStatus 1, $\pm$ 0x03 AND ringStatus 2, $\pm$ 0x03	
	DELAV 1 Second	
	CET ringStatus 1 ringStatus 2	
	WEND	
	No (a la angli angli angli angli ang	
	Note: Loop until controller rests in green somewhere.	
	Set pnaseControlGroupVehCall.1 = 0x80	
	DELAY .2 Seconds	
	Set phaseControlGroupVehCall.1 = 0x00	
	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	

	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x81 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
	DFLAY 1 Second	
	GET nhaseStatusGrounPhaseOns 1	
	nhaseStatusGrounPhaseOns 2	
	VVEND Noto Wait until controller reaches 1 / 9	
	Set HITL Detector Input 4 = On	
	DELAY 1.9 Seconds	
	GET phaseStatusGroupVehCalls.1	
	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =	Pass/Fail
	0x00	
	Note: Verify that detector call is not active yet	
	DELAY 2 Seconds	
	CET phaseStatusCroupVahCalls 1	
		D/E "
	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =	Pass/Fail
	0x08	
	Note: Verify that detector call is now active.	
	Set HITL Detector Input 4 = Off	
Detector 6 Delay		•
Detector 6 Delay	GET ringStatus.1, ringStatus.2	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere.	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere.	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00 GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00 GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00 GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND phaseStatusGroupPhaseOns.2 ≠ 0X00	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00 GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00 GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second CET phaseStatusGroupPhaseOns.1	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00 GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1,	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00 GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND <i>Note:</i> Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00 GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND	
Detector 6 Delay	GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x02         DELAY .2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2, we cond         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 2+5.	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00 GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WEND Note: Wait until controller reaches 2+5.	
Detector 6 Delay	GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x02         DELAY .2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2, we can be added by the second by the se	
Detector 6 Delay	GET ringStatus.1, ringStatus.2 WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03 DELAY 1 Second GET ringStatus.1, ringStatus.2 WEND Note: Loop until controller rests in green somewhere. Set phaseControlGroupVehCall.1 = 0x02 DELAY .2 Seconds Set phaseControlGroupVehCall.1 = 0x00 GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND phaseStatusGroupPhaseOns.2 ≠ 0X00 DELAY 1 Second GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2, WEND Note: Wait until controller reaches 2+5. Set HITL Detector Input 6 = On	
Detector 6 Delay	GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x02         DELAY .2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 2+5.         Set HITL Detector Input 6 = On         DELAY 1.9 Seconds	
Detector 6 Delay	GET ringStatus.1, ringStatus.2         WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03         DELAY 1 Second         GET ringStatus.1, ringStatus.2         WEND         Note: Loop until controller rests in green somewhere.         Set phaseControlGroupVehCall.1 = 0x02         DELAY .2 Seconds         Set phaseControlGroupVehCall.1 = 0x00         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.1 ≠ 0x12 AND         phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2         WHILE phaseStatusGroupPhaseOns.2 ≠ 0X00         DELAY 1 Second         GET phaseStatusGroupPhaseOns.2         WEND         Note: Wait until controller reaches 2+5.         Set HITL Detector Input 6 = On         DELAY 1.9 Seconds         GET phaseStatusGroupVehCalls.1	

	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00	Pass/Fail
	Note: Verify that detector call is not active yet.	
	DELAY .2 Seconds	
	GET phaseStatusGroupVenCalls.1	Dece/Feil
	0x20	Pass/Fail
	<i>Note:</i> Verify that detector call is now active.	
	Set HITL Detector Input 6 = Off	
Detector 7 Delay		
Detector / Detay	GET ringStatus 1 ringStatus 2	
	WHILE ringStatus 1 $\neq$ 0x03 AND ringStatus 2 $\neq$ 0x03	
	DELAY 1 Second	
	GET ringStatus.1. ringStatus.2	
	WEND	
	Note: Loop until controller rests in green somewhere.	
	Sat abaseCaptrolCrownV/abCall 4 = 0v04	
	DELAY .2 Seconds	
	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2 WHILE phaseStatusGroupPhaseOns.1 ≠ 0x14 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
	DELAY 1 Second	
	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
	WEND	
	<i>Note:</i> Wait until controller reaches 3+5.	
	Set HITL Detector Input 7 = On	
	DELAY 1.9 Seconds	
	GET phaseStatusGroupVehCalls.1	
	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x00	Pass/Fail
	Note: Verify that detector call is not active yet.	
	DELAY .2 Seconds	
	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 = 0x40	Pass/Fail
	Note: Verify that detector call is now active.	
	Set HITI Detector Input 7 - Off	

Detector 8 Delay		
	GET ringStatus.1, ringStatus.2	
	WHILE ringStatus.1 ≠ 0x03 AND ringStatus.2 ≠ 0x03	
	DELAY 1 Second	
	GET ringStatus.1, ringStatus.2	
	WEND	
	Note: Loop until controller rests in green somewhere.	
	Set phaseControlGroupVehCall.1 = 0x04	
	DELAY .2 Seconds	
	Set phaseControlGroupVehCall.1 = 0x00	
	GET phaseStatusGroupPhaseOns.1, phaseStatusGroupPhaseOns.2	
	WHILE phaseStatusGroupPhaseOns.1 ≠ 0x18 AND	
	phaseStatusGroupPhaseOns.2 ≠ 0X00	
	DELAY 1 Second	
	GET phaseStatusGroupPhaseOns.1,	
	phaseStatusGroupPhaseOns.2	
	WEND	
	Note: Wait until controller reaches 4+5.	
	Set HITL Detector Input 8 = On	
	DELAY 1.9 Seconds	
	GET phaseStatusGroupVehCalls.1	
	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =	Pass/Fail
	0x00	
	<i>Note:</i> Verify that detector call is not active yet.	
	DELAY .2 Seconds	
	GET phaseStatusGroupVehCalls.1	
	VERIFY that RESPONSE VALUE phaseStatusGroupVehCalls.1 =	Pass/Fail
	0x80	
	<i>Note:</i> Verify that detector call is now active.	
	Set HITL Detector Input 8 = Off	
Teardown		
	FOR DetectorNumber = 2, 3, 4, 6, 7, and 8	
	SET vehicleDetectorDelay.DetectorNumber =	Pass/Fail
	[currentDetectorDelay.DetectorNumber]	
	No (a. Destant the original value	
	Note: Restore the original value.	
-	NEX I Phase	
I est Case Results		
Tested By:	Date	Pass/Fail
	Tested	1 400,1 41
Test Case Notes:	<notes></notes>	
Version History:	v1.00 06/11/06 Initial Draft – RDR	
	v1.01 07/05/06 Updated notes – RDR	

## **REFERENCES FOR APPENDIX G**

- *1.* NTCIP Laboratory Testing for Actuated Signal Controllers, Summary Report for ASSHTO Project 475070. Published by Texas Transportation Institute. <u>http://tti.tamu.edu/documents/TTI-2006-1.pdf</u>. Accessed June 7, 2006.
- NTCIP 1202 Object Definitions for Actuated Traffic Signal Controller Units, A Joint Publication of AASHTO, ITE, and NEMA.
   <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=1202</u>. Accessed June 21, 2006.
- 3. Balke, K., R. Engelbrecht, S. Sunkari, J. Charara, TTI'S Hardware-In-The-Loop Traffic Signal Controller Evaluation System. Published by Texas Transportation Institute. <u>http://tti.tamu.edu/documents/5-1752-01-1.pdf</u>. Accessed August 23, 2006.
- 4. DMS-11170, Fully Actuated, Solid-State Traffic Signal Controller Assembly, Departmental Material Specifications 7-115, Section 19. Published by TxDOT. <u>http://manuals.dot.state.tx.us/dynaweb/colmates/dms/@ebt-link;?target=idmatch(s070019)</u>. Accessed July 29, 2005.
- 5. NTCIP 1201 Global Object Definitions. A Joint Publication of AASHTO, ITE, and NEMA.
   <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar d=1201</u>. Accessed July 25, 2005.
- 6. NTCIP 8007 Testing and Conformity Assessment Documentation within NTCIP Standards Publications, A Joint Publication of AASHTO, ITE, and NEMA. <u>http://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standar</u> <u>d=8007</u>. Accessed July 25, 2005.