SIEMENS

APPLICATION REFERENCE MANUAL

GEOGRAPHIC CONFIGURATION SUITE (GCS)

MAY 2008 (REVISED APRIL 2015)

DOCUMENT NO. SIG-00-05-13
VERSION B.1

Siemens Industry, Inc. Rail Automation
9568 Archibald Ave., Suite 100,
Rancho Cucamonga, California 91730
1-800-793-SAFE

PRINTED IN U.S.A.

PROPRIETARY INFORMATION

TRANSLATIONS

The manuals and product information of Siemens Industry, Inc. are intended to be produced and read in English. Any translation of the manuals and product information are unofficial and can be imprecise and inaccurate in whole or in part. Siemens Industry, Inc. does not warrant the accuracy, reliability, or timeliness of any information contained in any translation of manual or product information from its original official released version in English and shall not be liable for any losses caused by such reliance on the accuracy, reliability, or timeliness of such information. Any person or entity that relies on translated information does so at his or her own risk.

WARRANTY INFORMATION

Siemens Industry, Inc. warranty policy is as stated in the current Terms and Conditions of Sale document. Warranty adjustments will not be allowed for products or components which have been subjected to abuse, alteration, improper handling or installation, or which have not been operated in accordance with Seller's instructions. Alteration or removal of any serial number or identification mark voids the warranty.

SALES AND SERVICE LOCATIONS

Technical assistance and sales information on **Siemens Industry, Inc.** products may be obtained at the following locations:

SIEMENS INDUSTRY, INC. RAIL AUTOMATION
2400 NELSON MILLER PARKWAY
LOUISVILLE, KENTUCKY 40223

TELEPHONE:	(502) 618-8800
FAX:	(502) 618-8810
SALES & SERVICE:	(800) 626-2710
WEB SITE:	http://www.siemens.com/rail-automation

SIEMENS INDUSTRY, INC.RAIL AUTOMATION
939 S. MAIN STREET
MARION, KENTUCKY 42064

TELEPHONE:	(270) 918-7800
CUSTOMER SERVICE:	(800) 626-2710
TECHNICAL SUPPORT:	(800) 793-7233
FAX:	(270) 918-7830

DOCUMENT HISTORY

Version	Release Date	Sections Changed	Details of Change
A.8	MAY 2008	All	Initial Release as Preliminary
B	AUG 2013	- - -	Official Siemens release as version B (No change in content)
B.1	APR 2015	4	Sec 4: Pg 4-9 Para 4.16 (4.17 after 4.17 move) change atcslayout to atcsconfig. Pg 4-10, Para 4.16 (4.17 after 4.17 move) change BG to BL Para 4.17 moved in between 4.11 and 4.12

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTES, CAUTIONS, AND WARNINGS

Throughout this manual, notes, cautions, and warnings are frequently used to direct the reader's attention to specific information. Use of the three terms is defined as follows:

⚠ WARNING

WARNING

INDICATES A POTENTIALLY HAZARDOUS SITUATION WHICH, IF NOT AVOIDED, COULD RESULT IN DEATH OR SERIOUS INJURY. WARNINGS ALWAYS TAKE PRECEDENCE OVER NOTES, CAUTIONS, AND ALL OTHER INFORMATION.

⚠ CAUTION

CAUTION

REFERS TO PROPER PROCEDURES OR PRACTICES WHICH IF NOT STRICTLY OBSERVED, COULD RESULT IN A POTENTIALLY HAZARDOUS SITUATION AND/OR POSSIBLE DAMAGE TO EQUIPMENT. CAUTIONS TAKE PRECEDENCE OVER NOTES AND ALL OTHER INFORMATION, EXCEPT WARNINGS.

NOTE

NOTE

Generally used to highlight certain information relating to the topic under discussion.

If there are any questions, contact Siemens Industry, Inc. Application Engineering.

ELECTROSTATIC DISCHARGE (ESD) PRECAUTIONS

Static electricity can damage electronic circuitry, particularly low voltage components such as the integrated circuits commonly used throughout the electronics industry. Therefore, procedures have been adopted industry-wide which make it possible to avoid the sometimes invisible damage caused by electrostatic discharge (ESD) during the handling, shipping, and storage of electronic modules and components. Siemens Industry, Inc. has instituted these practices at its manufacturing facility and encourages its customers to adopt them as well to lessen the likelihood of equipment damage in the field due to ESD. Some of the basic protective practices include the following:

- Ground yourself before touching card cages, assemblies, modules, or components.
- Remove power from card cages and assemblies before removing or installing modules.
- Remove circuit boards (modules) from card cages by the ejector lever only. If an ejector lever is not provided, grasp the edge of the circuit board but avoid touching circuit traces or components.
- Handle circuit boards by the edges only.
- Never physically touch circuit board or connector contact fingers or allow these fingers to come in contact with an insulator (e.g., plastic, rubber, etc.).
- When not in use, place circuit boards in approved static-shielding bags, contact fingers first. Remove circuit boards from static-shielding bags by grasping the ejector lever or the edge of the board only. Each bag should include a caution label on the outside indicating static-sensitive contents.
- Cover workbench surfaces used for repair of electronic equipment with static dissipative workbench matting.
- Use integrated circuit extractor/inserter tools designed to remove and install electrostatic-sensitive integrated circuit devices such as PROM’s (OK Industries, Inc., Model EX-2 Extractor and Model MOS-40 Inserter (or equivalent) are highly recommended).
- Utilize only anti-static cushioning material in equipment shipping and storage containers.

For information concerning ESD material applications, please contact the Technical Support Staff at 1-800-793-7233. ESD Awareness Classes and additional ESD product information are also available through the Technical Support Staff.

GLOSSARY

AAR:	Association of American Railroads – An organization that establishes uniformity and standardization among different railroad systems.
AREMA:	American Railroad Equipment Manufacturing Association – An organization that supersedes AAR.
Aspect:	(Signal Aspect) The appearance of a fixed signal conveying an indication as viewed from the direction of an approaching train. A cab signal conveying an indication as viewed by an observer in the cab.
Aspect I/O:	The aspect input and output channel assignment to a search light or color light card. This corresponds to the aspect I/O configuration selection.
Aspect Representation:	Illustrated indication of a signal display. Each illustration represent a particular rule for an approaching train, as seen by an observer.
Aspect Set:	A set of different color combinations a particular signal displays at a given location. Sets are grouped by the number of heads the signal has, with three being the maximum number of heads allowed.
ATCS:	Advanced Train Control System – A set of standards compiled by the AAR for controlling all aspects of train operation.
Circuit I/O:	The VRO and VPI channel assignments of a GEO unit.
DTU:	Diagnostic Terminal Utility – The software used to download the GEO Module Configuration File (MCF) into the Geographic Signaling System (GEO) unit.
ECD:	External Configuration Device – A serial EEPROM (Flash Memory) device mounted inside the chassis of the GEO unit. The ECD is used to store site-specific configuration data (MCF, SIN, UCN, and card parameters) for the CPU.
GCS:	GEO Configuration Suite – The GEO Logic application software used to create and simulate track layouts. Then it produces a GEO MCF, which configures the GEO unit.

GLOSSARY

GEO Explorer Tab:	Workspace window that allows the user to modify and manage railroads, lines, and groups. Provides assignments for GEO chassis, edit I/O assignments, MCFs, and NVLPs to groups.
GEO MCF:	GEO Module Configuration File – The site-specific configuration information created by the GCS and downloaded into the ECD via the CPU and Diagnostic Terminal Utility (DTU).
GEO Module:	GEO PCB I/O card. Modules (cards) install in GEO Wayside Units and provide I/O channels to control GEO hardware.
GEO Wayside Unit:	GEO equipment used at a site location to operate railroad lights, switches or transmit information to other locations.
Geographic Object:	A graphic representation tile of a virtual track object. Typical objects are signal, coded track interface, switches, track, traffic, end of block and diamond.
Geographic Object I/O:	The input and output configuration that an object is selected to operate with.
GL Properties:	The various GEO Logic properties an object has which contain values. These values can be set to produce various operating conditions.
GL Script Tab:	The workspace window that allows the user to view the controls, inputs and outputs of a simulated script that the simulator runs.
GL Simulation:	The simulated actions that a track layout will exhibit during actual use.
GL Subroute Properties:	The subroute properties a geographic object can exhibit. An object can have multiple subroutes depending on the track layout. Each subroute of an object contains the same properties yet the values of the properties can be set differently to reflect that actual track configuration.
GL Table:	There are four element types of GEO Logic tables. Aspect, code, cab and circuit are the four types and a GEO Logic table can contain one or more of these element types.

GLOSSARY

GOL:	Geographic Object Library – A collection of Geographic Objects that appear on the Geographic Object tool bar. These objects represent certain track elements (coded track interfaces, signals, switches, etc.) that possess characteristics of those track elements as defined by the FRA rules.
Group:	A sub-header under the Line entry, to organize the information created by the GCS. The Group location is where the Chassis, NVLP and MCF's, of a track layout are stored. The Group entry has a number, name, state location and milepost that can be associated with ATCS addressing.
HDF:	Hardware Description File – A utility file for configuring a module and reading status and diagnostic information.
I/O Card:	Circuit cards contained within the GEO module. These cards pass input and output information to tracks, signals, and other railroad equipment.
I/O Editor Tab:	The workspace window that allows the user to select the input and output hardware channel assignments for the circuit, cab, code, and aspect GEO logic types.
Line:	A sub-header under Railroad to organize the information created by the GCS. The Line entry has a number, name and description that can be associated with ATCS addressing.
MEF:	Module Executable File – The executive software running in the CPU. The user can download the MEF through the DTU port to update the software.
Menu Path:	The logical path from the GEO menu bar to appropriate sub menus for completing a task.
NVLP:	Non-Vital Link Processor – Processor that connects non-vital circuits to the GEO unit.
Railroad:	The main header for the information created by the GCS. The Railroad has a number, mnemonic and name that can be associated with ATCS addressing.
Route Signature:	The subroute name that assigns the property value information of a particular route to an object.

GLOSSARY

Simulator I/O View Tab: The workspace window that allows the user to view the controls, indications, inputs and outputs of a geographic object.

SIN: Site Identification Number – The 12-digit ATCS address for the GEO equipment, downloaded through the Diagnostic serial port on the CPU and stored in the ECD. The SIN has the form 7.RRR.LLL.GGG.SS stored in binary coded decimal, with each digit in one nibble. The digit 0 is represented by "A" and 0 is used as a null byte.

Site Location: The location where GEO wayside unit is installed, usually within close proximity of the signal or switch, that the GEO unit is controlling.

Stop Aspect: The most restrictive aspect of an aspect set. The aspect that appears at the end of a simulation.

Subroute: A route that has a beginning and an end. During configuration of an object the subroute of the object is highlighted in green.

Subroute Properties: Property values that are specifically set for a particular subroute. Additional subroutes can be set to contain the same property values or set to have different values.

UCN: Unique Check Number – A 32-bit CRC calculated over the MCF and stored in the ECD to detect file corruption. The UCN is created by the GCS and entered into the CPU to verify proper configuration.

VPI: Vital Parallel Input – A module input circuit the function of which affects the safety of train operation.

VRO: Vital Relay Output – A module output circuit the function of which affects the safety of train operation.

SECTION 1
INTRODUCTION

1.0 INTRODUCTION

The Siemens GEO Configuration Suite (GCS) is a powerful Windows®-based computer application used to design interlockings based on the various components of Siemens GEO System. The GCS uses Siemens GEO/Logic to model interlockings without the traditional hassles associated with relay circuit design and ladder logic or Boolean expressions. This tool allows railroad signal designers to easily create track layouts that mimic track plans and run simulations for track layout.

The primary components of the GCS user interface are *Geographic Objects* that represent specific parts of a track plan such as signals, switches and track elements. A symbol on each Geographic Object identifies the track plan component. The basic Geographic Objects are preprogrammed with the applicable AREMA signaling rules for the device or track element they represent. Using the GCS, the signal designer can further configure these Geographic Objects to reflect specific railroad requirements such as signal *Aspects*.

1.1 INCLUDED TUTORIAL

If you're new to the GCS application, a tutorial is included in this document in Section 4. This tutorial provides the introductory knowledge required to get you started by designing a basic intermediate application. Once you have an understanding of how the GCS application works, the programming information presented in Sections 2 and 3 provide the details to create, build and test much more sophisticated layouts.

1.2 COMPUTER SYSTEM REQUIREMENTS

The list below indicates the recommended computer configuration for the GCS Programming Application.

- VGA monitor.
- Pentium 4® or equivalent processor, 1.5 GHz or faster.
- 512 MB of RAM.
- Hard disk with sufficient free space to install the required application and maintain files (approximately 300 MB minimum).
- CD-ROM drive for application installation.
- Microsoft® Windows® Operating System version 2000, XP or later.

1.3 ORDERING INFORMATION

Siemens part numbers for GCS Programming Applications:

- GCS Z224-9V887-A010
- GCS with UCN Calculator Z224-9V888-A010

1.4 PROGRAM INSTALLATION

1. Insert the GCS installation disk into your computer's CD ROM drive.
2. After the CD spins up, the installation startup screen should automatically appear. If the installation screen does not appear within a short time, browse the CD using My Computer or Windows Explorer and locate the file named Setup.exe. Double-click the Setup file to begin the installation process.
3. Follow the on-screen instructions to install the application(s).

1.5 MCF VERIFICATION AND VALIDATION

Verification and Validation (V&V) of the MCF, as with all software, requires code level and system level testing. The GCS includes tools that facilitate lower level V&V activities and Siemens recommends a V&V approach utilizing a combination of Simulation, Decompilation and Scripting. A system level approach, such as rack testing is a required V&V activity, based on the system safety analysis.

⚠ WARNING

WARNING

BEFORE BEING PLACED IN LIVE-SERVICE, THE MCF GENERATED USING THE GCS MUST BE VERIFIED AND VALIDATED TO ENSURE SAFETY.

- **Simulation (Recommended)**

Interactive simulation may be used to test the logical operation of the application. All logical combinations of input states may be explicitly commanded and all resulting output states may be inspected using the GCS Track Layout view and DT System States view to ensure expected operation of configuration data and vital logic (from both pre-defined GOL and user defined logic).

- **Decompilation (Recommended)**

  Module Configuration Files are compiled from user supplied GC and GL files. Decompilation may be used to re-generate the original user-supplied input files (both GC and GL files) from compiled Module Configuration Files - decompilation will also automatically re-generate the GL files for the pre-defined GOL that was originally used to compile the Module Configuration Files. Comparing the original GC and GL files with the decompiled GC and GL files is one way to ensure that all configuration data and vital logic are accurately reflected in compiled Module Configuration Files. A Module Configuration File (MCF) must be decompiled to obtain an MCF CRC which is required to place the MCF in service.

- **Scripting (Recommended)**

  Scripting may be used to formalize interactive simulation testing to test the logical operation of an application by documenting input states and expected output states in script files. Scripts may be re-used to test an application as it evolves. All logical combinations of input states may be explicitly commanded from a script and all resulting output states may be explicitly verified in a script to ensure expected operation of configuration data and vital logic (from both pre-defined GOL and user defined logic).

- **System Test (e.g. Rack Test) (Required)**

  Verifies the resultant MCF against actual hardware and executive software. As part of the system test, the Program Report obtained from the DT must be generated and reviewed to ensure all configuration parameters and default settings are accurately represented.

  The person performing the system test must be familiar with the GOL and the software application used.

This Page Intentionally Left Blank

SECTION 2
GEO CONFIGURATION LISTING REFERENCE

2.0 GEO CONFIGURATION LISTING REFERENCE

This section describes the syntax and construction of a GEO Configuration Listing used to specify a GEO application program.

2.1 SYNTAX NOTATION FORMATS

NOTATION TYPE	FORMAT
Syntactic categories	*Italic* type
Literal words and characters	**Bold** type
Optional items	Enclosed in brace brackets or subscripted with *opt*
Repeating items	The characters … are used

2.2 GEO CONFIGURATION LISTING CONTENTS

A GEO Configuration Listing consists of one or more text files; each ending in .gc extension (e.g., eos.gc).

2.2.1 Components

These may be defined in one or more GC files:

- Aspect chart definition.
- Aspect set and degraded aspect table definitions.
- Table sets.
- A track layout.
- Logical configurations.
- ATCS configurations.
- GWE (GEO Wayside Equipment) configurations.

2.2.2 Comments

The # character introduces a comment.

2.2.3 Identifiers

An identifier is a sequence of letters and digits; the underscore _ counts as a letter. Identifiers are case-sensitive, and may have any length. Referred to in this document as *identifier*.

2.2.4 Keywords

The following identifiers are reserved for use as keywords, and may not be used otherwise. Note that keywords are case-sensitive in that lowercase keywords are "qualifiers".

ACTIVATE	**AS**	**ASPECT**
aspect	**ASPECTCHART**	**ASPECTDEF**
ASPECTSET	**aspectset**	**ASSIGN**
ATCSCONFIG	**atcsdevice**	**atcslayout**
bit	**connec**	**context**
CONTROL	**defaultvalue**	**DEGASPECT**
desc	**destination**	**DEVICE**
enable	**enum**	**fieldconfig**
GROUP	**gwe**	**GWECONFIG**
hflip	**INDICATION**	**INTCH**
integer	**integer1**	**integer2**
LINE	**LINK**	**LOGICALCONFIG**
logicallayout	**MODULE**	**module**
number	**OBJECT**	**RAILROAD**
rotation	**route**	**searchlight**
slot	**stopaspect**	**SUBNODE**
subnode	**TABLE**	**TABLESET**
TO	**TRACKLAYOUT**	**tracklayout**
type	**vflip**	**VMODULE**
xpos	**ypos**	

2.2.5 Constants

All constants are unsigned integer values consisting of a sequence of digits. Referred to in this document as *integer-constant*. Identifiers declared as enumerators are considered to be integer constants.

2.2.6 Strings

A string is a sequence of characters surrounded by double quotes. Referred to in this document as *string*.

2.2.7 Aspect Representation

A sequence of characters (representing colors) delimited by a forward slash (/) to identify signal heads. Use to identify a physical representation of a signal (e.g., G/R, G/D). Valid characters representing colors are G for green, Y for yellow, R for red, L for lunar and D for dark. Referred to in this document as *aspect-rep*.

2.3 ASPECT CHART

An aspect chart is used to define aspects and aspect representations that will be referenced by the application. An aspect is a signal indication such as Clear or Approach. An aspect representation is the physical make up of the signal indication such as G/R for Clear or Y/R for Approach. Each GEO application may specify zero or one aspect chart.

An aspect chart has the following form:

aspect-chart

ASPECTCHART *name { **[** description-qualifier **]** }* **(** *aspect-def {, aspect-def, …}* **)**

name may be any identifier

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the aspect chart.

aspect-def

ASPECTDEF *aspect-name { **[** description-qualifier **]** } { **(** aspect-rep {, aspect-rep, …}* **)** *}*

aspect-name may be any *identifier* or a *string* (use a string if aspect name contains spaces). This name will be used to reference the aspect.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the aspect.

Example:

```
ASPECTCHART Sample [desc="Sample Aspect Chart"]
(
      ASPECTDEF CLEAR (G/R, G/D, G),
      ASPECTDEF "ADVANCE APPROACH" (FY/R, FY/D, FY),
      ASPECTDEF APPROACH (Y/R, Y/D),
      ASPECTDEF RESTRICTING (R/Y, D/Y, Y),
      ASPECTDEF STOP (R/R, R),
)
```

An aspect chart must specify at least one aspect definition. An aspect definition may specify none, one or more aspect representations. An aspect definition must include all degraded aspect representations (such as G/D).

For coded track applications, it is recommended that “dummy” aspects be defined for each vital code to facilitate specification of code and aspect configuration in later steps.

Example:

```
ASPECTCHART Sample [desc="Sample Aspect Chart (with aspects
representing vital codes)"]
(
      ASPECTDEF CLEAR (G/R, G/D, G),
      ASPECTDEF "ADVANCE APPROACH" (FY/R, FY/D, FY),
      ASPECTDEF APPROACH (Y/R, Y/D),
      ASPECTDEF RESTRICTING (R/Y, D/Y, Y),
      ASPECTDEF STOP (R/R, R),
      ASPECTDEF CODE7,
      ASPECTDEF CODE4,
      ASPECTDEF CODE3,
      ASPECTDEF CODE2,
      ASPECTDEF CODE8,
      ASPECTDEF CODE9
)
```

If an aspect chart is defined in full in its own GC file for a particular railroad based on that railroads timetable or rule book then that GC file may be reused for all applications on that railroad.

2.4 ASPECT SETS AND DEGRADED TABLES

An aspect set is a subset of “compatible” aspect representations from the aspect chart consisting of all aspects (including degraded aspects) that a particular signal would be expected to display. None, one or more aspect sets may be defined. Optionally, a degraded aspect table may be associated with an aspect set.

An aspect set together with an associated degraded aspect table has the following form:

aspect-set-and-degraded-aspect-table
    aspect-set { degraded-aspect-table }

2.4.1 Aspect Set

aspect-set has the following form:

aspect-set

ASPECTSET *name { **[searchlight**$_{opt}$, description-qualifier$_{opt}$ **]** }* **(** *aspect-set-entry {, aspect-set-entry, …}* **)**

name may be any *identifier*. This name will be used to reference the aspect set.

searchlight

Qualify aspect set with the **searchlight** keyword to designate that the aspect set is intended for a searchlight signal; omit **searchlight** keyword if the aspect set is intended for a colorlight signal.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the aspect set.

aspect-set-entry

aspect-rep { **[aspect =** *aspect-name {,* **stopaspect** *}* **]** *}*

aspect-set-entry specifies an aspect representation from the aspect chart.

aspect = *aspect-name*

If *aspect-rep* is ambiguous because the aspect chart defines multiple aspects with the same aspect representation then use the aspect keyword to qualify the aspect representation with the aspect name (e.g., G/R **[aspect =** Clear **]**). *aspect-name* may be any *identifier* or a *string* (use a string if aspect name contains spaces) that identifies an aspect from the aspect chart.

stopaspect

One of the aspect set entries in an aspect set must be designated as the most restrictive aspect by qualifying the aspect representation with the **stopaspect** keyword.

If multiple representations of the same aspect are assigned to an aspect set, then one of the representations must be designated as the primary representation (the other representations will be referred to as secondary representations). A primary aspect representation may be any representations (i.e., with or without dark heads); secondary aspect representations must have at least one dark head.

The aspect representations in an aspect set that represent the same aspect are enumerated such that the primary aspect representation is enumerated as zero. Multiple aspect representations that represent the same aspect must be compatible with respect to the primary aspect representation in order to be assigned to the same aspect set.

A primary aspect representation and a secondary aspect representation are compatible if corresponding heads are the same color and the secondary aspect representation has at least one dark head. Also, if there are any dark heads on the primary aspect representation then the corresponding head(s) on the secondary aspect representation must also be dark. For example: G/R/R is compatible with G/R/D; G/R/R is not compatible with Y/R/R; G/R/R is not compatible with G/Y/D; G/R/R is compatible with G/D/D; G/D is not compatible with D/G.

Example:

```
ASPECTSET FaTwoHead [searchlight, desc="Facing Two Head Aspect Set"]
(
      G/R, G/D,
      FY/R, FY/D,
      Y/R, Y/D,
      R/Y, D/Y,
      R/R [aspect=STOP, stopaspect]
)
```

2.4.2 Degraded Aspect Table

Optionally, one and only one degraded aspect table may be associated with an aspect set. A degraded aspect table specifies which aspect to display if the desired aspect cannot be displayed due to lamp-out conditions.

degraded-aspect-table has the following form:

degraded-aspect-table

DEGASPECTTABLE [aspectset = *aspect-set-name, description-qualifier$_{opt}$* **] (**
degraded-aspect-table-entry {, degraded-aspect-table-entry, …} **)**

aspectset = *aspect-set-name*

Use **aspectset** keyword to identify the associated aspect set. *aspect-set-name* may be any *identifier* that identifies an aspect set.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the degraded aspect table.

A degraded aspect table may have one or more degraded aspect table entries. The number of degraded aspect table entries is limited by the number of aspect representations specified in the aspect set associated with the degraded aspect table.

The *degraded-aspect-table-entry* specifies two aspect representations – both aspect representations must be specified in the aspect set associated with the degraded aspect table. The first aspect representation identifies the desired aspect to be displayed; the second aspect representation identifies the aspect to be displayed if the first aspect representation cannot be displayed due to lamp-out conditions. The first aspect representation of all degraded aspect table entries in a degraded aspect table must be unique (i.e., the first aspect representation in the set of degraded aspect table entries cannot repeat an aspect representation from the aspect set associated with the degraded aspect table).

degraded-aspect-table-entry

(aspect-rep { [**aspect** = aspect-name] } {, aspect-rep { [**aspect** = aspect-name] } })

aspect = aspect-name

If aspect-rep is ambiguous because the aspect chart defines multiple aspects with the same aspect representation then use the aspect keyword to qualify the aspect representation with the aspect name (e.g., G/R [**aspect** = Clear]).

2.5 TABLE SET

Use a table set to predefine tables that will be assigned to multiple geographic objects.

A table set has the following form:

table-set

TABLESET *name* { [*description-qualifier*] } (*table* {*, table, ...* })

name may be any *identifier*

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the Table Set.

table

TABLE *name* { [*description-qualifier*] } (*table-column* {*, table-column* }) = (*table-entry* {, *table-entry*, ... })

name may be any *identifier* specifying the name of the table.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the table.

table-def-qualifiers

/* optional */
'**[**' *table-def-qualifier_list* '**]**'

table-def-qualifier_list

table-def-qualifier
table-def-qualifier_list ',' table-def-qualifier

table-def-qualifier
    description-qualifier

table-column-list
    table-column
    table-column-list ',' table-column

table-column
    identifier table-column-qualifiers

table-column-def
    name [{ *table-column-def-qualifier*, *table-column-def-qualifier*, ... }]
    name may be any *identifier*

table-column-def-qualifier
    aspect-id-qualifier
    enum-qualifier
    context-qualifier
    default-value-qualifier
    description-qualifier

aspect-id-qualifier
    ASPECT

    aspect-id-qualifier specifies that a property or route property may be any integer that represents a valid Aspect definition.

enum-qualifier
    enum = *name*

    enum-qualifier specifies that a property or route property may be any enumerator from the enumeration identified by *name*.

context-qualifier
    context = *context-name*

    context-name identifies an enumeration or context that will further restrict the range of values that the property may represent.

default-value-qualifier

    defaultvalue = *integer-constant*

    defaultvalue = *name*

    integer-constant must be a value consistent with the range/type specified by *size-qualifier*.

    name must be an enumerator associated with the enumeration specified by *enum-qualifier*.

description-qualifier
    desc = *string*

    description-qualifier specifies an optional text string describing the table column.

table-entry

value

(*left-value, right-value*)

table-entry represents a row in a table; *value* specifies a single column row; *left-value*, *right-value* specifies a two column row. *value*, *left-value*, and *right-value* must be consistent with corresponding table column definition.

table-entry-def-qualifier_list

table-entry-def-qualifier

table-entry-def-qualifier_list ‘,’ *table-entry-def-qualifier*

table-entry-def-qualifier

description-qualifier

field_config_qualifier

gl-value-pair

'(' *gl-value ',' gl-value* ')'

gl-value-list

gl-value

gl-value-list ',' gl-value

gl-value

CONSTANT

identifier

name

qualified_aspect-rep

2.6 TRACK LAYOUT

A track layout consists of a set geographic objects, the geographic connections between them, and the list of I/O that is to be activated for the application. Each GEO application must specify one and only one track layout specification.

A track layout has the following form:

track-layout

TRACKLAYOUT *name { **[***description-qualifier* **]** *}* **(**

object-list

connection-list$_{opt}$

active_io_list$_{opt}$

)

name may be any identifier.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the track layout.

2.6.1 Object List

object-list specifies a list of geographic objects, their assigned behaviors from the Geographic Object Library and their graphical characteristics.

object-list has the following form:

object-list

object {, object, … }

object

OBJECT *name* **[** *description-qualifier*$_{opt}$ *type-qualifier x-pos-qualifier y-pos-qualifier rotation-qualifier*$_{opt}$ **hflip vflip** *atcs-device-qualifier*$_{opt}$ **]**

name may be any *identifier*. This name will be used to reference the object.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the object.

type-qualifier

type = name

type-qualifier specifies the type of the object; *name* is an *identifier* that identifies a SAT from a Geographic Object Library.

x-pos-qualifier

xpos = *constant*

x-pos-qualifier specifies the x-coordinate of the object in the track layout diagram.

y-pos-qualifier

ypos = *constant*

y-pos-qualifier specifies the y-coordinate of the object in the track layout diagram.

rotation-qualifier

rotation = *constant*

rotation-qualifier specifies the rotation of the object; *constant* may be 0, 90, or 180 degrees. Default is zero degrees if not specified.

hflip

Use **hflip** keyword to horizontally flip an object in the track layout view.

vflip

Use **vflip** keyword to vertically flip an object in the track layout view.

atcs-device-qualifier

atcsdevice = *constant*

atcs-device-qualifier specifies the ATCS device number of the object. If omitted then an ATCS device number will be automatically assigned (in sequential, ascending order starting at 4).

Example:

The following is a track layout specification for an intermediate signal application:

```
OBJECT WTK [type=CTIntf, xpos=80, ypos=80, rotation=0]
OBJECT WG [type=Signal, xpos=120, ypos=160, rotation=0]
OBJECT EG [type=Signal, xpos=160, ypos=160, rotation=180]
OBJECT ETK [type=CTIntf, xpos=200, ypos=160, rotation=180]
```

2.6.2 Connection List

connection-list specifies a list of geographic connections – the geographic connections between the geographic objects in the track layout.

connection-list has the following form:

connection-list

connection {, connection, … }

connection

LINK *object-name* **.** *connection-name* **TO** *object-name* **.** *connection-name*

connection specifies a geographic connection between two geographic objects.

object-name is an *identifier* that identifies a geographic object from the track layout.

connection-name is an *identifier* that identifies a geographic connection of the geographic object specified by *object-name*.

Example:

```
LINK WTK.ba TO WG.he
LINK WG.ba TO EG.ba
LINK EG.he TO ETK.ba
```

2.6.3 Active I/O List

All inputs and outputs that will be assigned to physical input/output channels must be activated using the **ACTIVATE** statement. Those inputs/outputs that will be mapped to CTC control and indications bits need not be activated.

active-sat-instance-io-list

{ *active-sat-instance-io,* { *active-sat-instance-io* } }

active-sat-instance-io

ACTIVATE *object-name* (*io-name* { **AS** *type* } {, *io-name* { **AS** *type* } })

object-name is an identifier that specifies an object from the track layout.

io-name is an identifier that specifies an input or an output of the object.

2.7 LOGICAL CONFIGURATION

A logical configuration defines the individual configuration of each geographic object in the track layout. Each GEO application may specify one or more logical configurations.

A logical configuration has the following form:

logical_config

LOGICALCONFIG *name { * **[** *description-qualifier* **]** *}* **(** *object-config-list* **)**

name may be any *identifier* that will be used to identify the logical configuration.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the logical configuration.

2.7.1 Object Configuration List

object-config-list specifies a list of geographic objects from the track layout and associated configuration data (the content of which is determined by the user and the nature of which is determined by both the behavior assigned to the object from the Geographic Object Library and the track layout).

object-config-list has the following form:

object-config-list

object-config {, object-config, … }

object-config

OBJECT *name {* **[** *connection_qualifier*$_{opt}$**,** *route_qualifier* **]** *}* **(** *config_item { config_item … }* **)**

object-config specifies the configuration data associated with a particular object. If an object is associated with one or more routes then use a separate *object-config* statement to specify route specific configuration information for each route.

name is an *identifier* that specifies a geographic object from the track layout.

Omit *connection_qualifier* and *route_qualifier* to specify SAT specific configuration data. If the *object-config* statement is to specify route specific configuration data then use *connection_qualifier* and *route_qualifier* to identify a specific route.

connection_qualifier

connec = *connec_name*

connection_qualifer specifes a connection of the geographic object that is designated by the Geographic Object Library as the start of a route.

connec_name is an *identifier* that specifies a connection of the geographic object.

route_qualifier

route = *object_name* **/** *path_name {* **,** *object_name* **/** *path_name* **,** *… }*

route_qualifier is the “name” of a specific route. *connection_qualifer* may be omitted if the object has only one connection that is designated as the start of a route.

object_name is an *identifier* that specifies a geographic object; *path_name* is an *identifier* that specifies a path defined by the object in the Geographic Object Library.

For example, the following identifies a route starting from the head connection of signal 1E containing SW1 in the normal position and switch 3W in the reverse position (switches must be specified in the order in which they would be encountered by a train moving on the route). In this example, the connection identifier (**connec**) is redundant and may be omitted since the head connection of a signal is the only connection that is designated as the start of a route in the Geographic Object Library.

```
OBJECT 1E [connec=he, route=1W/N, 3W/R]
```

config_item

*config_*item specifies a configurable item and an appropriate setting for that item.

config_item may specify the assignment of an aspect set.

For example, the following statement assigns an aspect set called TWO_HEAD_TRAILING to a Signal object (please note that AspectSet is an *identifier* specified by the Signal object in the Geographic Object Library).

```
AspectSet = TWO_HEAD_TRAILING
```

config_item may specify the setting of any property (SAT specific or route specific) defined by the type of the object in the Geographic Object Library.

For example, ApproachLocking – a configurable property of a Signal object – is disabled in the following statement:

```
ApproachLocking = ENABLETYPE_DISABLED
```

Any property may be made field-configurable with the **enable** keyword. When a property is made field-configurable, it shows up in the diagnostic terminal as a configurable option. For example:

```
ApproachLocking [enable] = ENABLETYPE_DISABLED
```

ResAspect – a configurable property of a Signal object – is set to a restricting aspect in the following example:

```
ResAspect = FR/R [aspect=Restricting]
```

config_item may specify the assignment of a previously defined table to an object. For example, a normal aspect table called Main (previously defined in the .gc file) is assigned to NorAspect (which is defined by a Signal object in the Geographic Object Library). Note, a colon and equal sign are required when referencing a predefined table.

```
NorAspect := Main
```

Objects which have multiple tables assigned to them automatically become field-configurable. The first table that is specified will be the default table. For example, a normal aspect table called Main1 and a normal aspect table called Main2 can be assigned to NorAspect as shown below. The default table will be Main1.

```
NorAspect := Main1, Main2
```

config_item may also specify an in-line assignment of a table. For example:

```
NorAspect = (
(CODE7, G/R),
(CODE4, FY/R),
(CODE3, R/R),
(CODE2, R/R),
(CODE8, Y/R),
(CODE9, R/R)
)
```

2.8 ATCS CONFIGURATION

An ATCS configuration specifies the assignment of geographic objects from the track layout into subnodes (i.e., GEO wayside units) and the location of the subnodes in terms of railroad number, line number, and group number. One or more ATCS configurations may be specified.

An ATCS configuration has the following form:

atcs_layout

ATCSCONFIG *name {* **[***description-qualifier* **]** *}* **(** *atcs_node_set* **)**

name may be any *identifier*. This name will be used to reference the ATCS configuration.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the ATCS configuration.

atcs_node_set

atcs_node_branch {, atcs_node_branch, … }

atcs_node_set specifies the ATCS organization of geographic objects in a track layout.

atcs_node_branch specifies one branch of a complete *atcs_node_set*.

atcs_node_branch

atcs_node { **(** *atcs_node_branch* **)** *}*

atcs_node_branch is recursively defined.

atcs_node

atcs_node_type atcs_node_name **[number =** *constant* **]**

atcs_node_name may be any *identifier*. This name will be used to reference the ATCS node.

atcs_node_type

RAILROAD

LINE

GROUP

SUBNODE

DEVICE

atcs_node_type specifies type of ATCS node.

number = ***constant***

number is a keyword to distinguish the ATCS node (i.e., to specify railroad, line, group, or subnode number).

The following examples illustrate various uses of subnodes:

Example 1: Single-unit EOS with railroad, line, group and subnode numbers specified explicitly.

```
ATCSCONFIG EOS
(
      RAILROAD GGR_RA [number=100] (
            LINE GGR_LI [number=100] (
                  GROUP GGR_GR [number=100] (
                        SUBNODE GGR_SU1 [number=3] (
                              DEVICE BT [number=1]
                              DEVICE FA_SIG [number=2]
                              DEVICE OST_1W [number=3]
                              DEVICE TN_SIG [number=4]
                              DEVICE TR_SIG [number=5]
                              DEVICE MT [number=6]
                              DEVICE MEL [number=7]
                              DEVICE BEL [number=8]
                              DEVICE BEOB [number=9]
                              DEVICE MEOB     [number=10]
                              DEVICE ST [number=11]
                        )
                  )
            )
)
)
```

Example 2: Single-unit EOS

```
ATCSCONFIG EOS
(
      SUBNODE GGR_SU1 [number=3] (
            DEVICE BT [number=1]
            DEVICE FA_SIG [number=2]
            DEVICE OST_1W [number=3]
            DEVICE TN_SIG [number=4]
            DEVICE TR_SIG [number=5]
            DEVICE MT [number=6]
            DEVICE MEL [number=7]
            DEVICE BEL [number=8]
            DEVICE BEOB [number=9]
            DEVICE MEOB     [number=10]
            DEVICE ST [number=11]
      )
)
```

Example 3: Distributed EOS: two subnodes in the same group (group is not explicitly identified)

```
ATCSCONFIG Distributed_EOS
(
      SUBNODE GGR_SU1 [number=3] (
            DEVICE BT [number=1]
            DEVICE FA_SIG [number=2]
            DEVICE OST_1W [number=3]
      )
      SUBNODE GGR_SU1 [number=3] (
            DEVICE TN_SIG [number=4]
            DEVICE TR_SIG [number=5]
            DEVICE MT [number=6]
            DEVICE MEL [number=7]
            DEVICE BEL [number=8]
            DEVICE BEOB [number=9]
            DEVICE MEOB     [number=10]
            DEVICE ST [number=11]
      )
)
```

Example 4: Distributed EOS: two subnodes in the same group (group is explicitly identified)

```
ATCSCONFIG Distributed_EOS
(
      GROUP GGR_GR [number=100] (
            SUBNODE GGR_SU1 [number=3] (
                  DEVICE BT [number=1]
                  DEVICE FA_SIG [number=2]
                  DEVICE OST_1W [number=3]
            )
            SUBNODE GGR_SU1 [number=3] (
                  DEVICE TN_SIG [number=4]
                  DEVICE TR_SIG [number=5]
                  DEVICE MT [number=6]
                  DEVICE MEL [number=7]
                  DEVICE BEL [number=8]
                  DEVICE BEOB [number=9]
                  DEVICE MEOB     [number=10]
                  DEVICE ST [number=11]
            )
      )
)
```

Example 5: Distributed EOS: two subnodes, each in a different group

```
ATCSCONFIG Distributed_EOS
(
      GROUP GGR_GR [number=100] (
            SUBNODE GGR_SU1 [number=3] (
                  DEVICE BT [number=1]
                  DEVICE FA_SIG [number=2]
                  DEVICE OST_1W [number=3]
            )
      )
      GROUP GGR_GR [number=100] (
            SUBNODE GGR_SU1 [number=3] (
                  DEVICE TN_SIG [number=4]
                  DEVICE TR_SIG [number=5]
                  DEVICE MT [number=6]
                  DEVICE MEL [number=7]
                  DEVICE BEL [number=8]
                  DEVICE BEOB [number=9]
                  DEVICE MEOB     [number=10]
                  DEVICE ST [number=11]
            )
      )
)
```

Although an ATCS configuration specifies explicit node numbers, only the difference between node numbers is relevant. During field configuration, any ATCS may be specified for any GWE as long as the relative node numbers between two ATCS addresses are consistent with the node numbers specified in the ATCS configuration.

2.9 GWE CONFIGURATIONS

GWE configuration describes the physical configuration of a GEO wayside unit. One GWE configuration is required for each GEO Wayside unit.

A GWE configuration has the following form:

gwe_configuration

> **GWECONFIG** name **[atcsconfig =** identifier, **subnode =** identifier, **gwe =** identifier, description-qualifier **] (** module_config io_configuration controls indications **)**
>
> *name* may be any *identifier*.

atcsconfig = *identifier*

> **atcsconfig** is a keyword to specify an ATCS configuration.

subnode = *identifier*

> **subnode** is a keyword to used specify a subnode from the specified ATCS configuration.

gwe

gwe is a keyword used to specify a GEO Wayside Unit. Valid GEO wayside units are as follows: TwoTrackExtended, ThreeTrackExtended.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the GEO wayside equipment configuration.

2.9.1 Module Configuration

Used to assign and configure GEO modules in a GEO wayside unit. Specify a module configuration statement for each slot of a GEO wayside unit that is to contain a GEO module.

A module configuration statement has the following form:

module_config

MODULE name **[slot =** *constant*, **module =** *identifier*, *description-qualifier*$_{opt}$ **]** *{* **(** *config_param {, config_param, …}* **)** *}*

name may be any *identifier* used to reference the GEO module

slot

slot is a keyword used to specify the slot number of the GEO module.

module

module is a keyword used to specify the module type.

module = *identifier*

Valid module types are as follows: VLP2, CodedTrack, CodedLine, VPI, VRO, Searchlight, Colorlight, RIO.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the GEO module.

config_param

name **=** *value units*

config_param specifies the default setting for a card configuration parameter.

name is any *identifier* that specifies a module configuration parameter for the relevant GEO module.

value is a constant or an identifier representing an enumerator (*value* must be valid for specified module configuration parameter).

units specifies units (required if parameter has units; otherwise omit).

```
MODULE sl1 [slot=1, module=VLP2] ( )
MODULE sl2 [slot=2, module=CodedTrack] (
      NonvitalCodeChangeCycles = 2,
      VitalCodeChangeCycles = 2,
      ShuntDropCycles = 2,
      ShuntPickCycles = 2
)
```

The following tables specify valid input/output channels and valid configuration/operating parameters for each type of GEO module:

2.9.1.1 Searchlight

Output channels: VRO 1
Input channels: VPI 1-2
Bidirectional channels: VLO 1-2

Operating Parameters:

PARAMETER NAME	VALUE (range/options)	DESCRIPTION
LampVoltage	**9000** thru **13000 mV** (by 20's)	Set Lamp Voltage. (millivolts) (adjustable in 20 mV increments)
VPIDebounce	**20** thru **200 ms** (by 2's)	Set VPI Debounce time. (milliseconds) (adjustable in 2 ms increments)
LampVoltageReg	Constant or Variable	Set to Constant to maintain constant lamp output voltage even under low battery conditions.
PCO1 Correspondence Time	**300** thru **3000 ms** (by 15's)	Set time allowed to detect correspondence between command position of searchlight mechanism 1 and the current position before declaring the mechanism as failed. (milliseconds)
PCO2 Correspondence Time	**300** thru **3000 ms** (by 15's)	Set correspondence detect time for searchlight mechanism 2. (see above)

Configuration Parameters:

None

2.9.1.2 Coded Track

Output channels: LED 1, VRO 1
Bidirectional channel: CTIO 1

Operating Parameters:

PARAMETER NAME	VALUE (range/options)	DESCRIPTION
VCOVoltage2	**0** thru **4000 mV** (by 20's)	Set coded track transmit voltage. (millivolts) (adjustable in 20mV increments)
CurrentLimit	**1000** thru **10000 mA** (by 50's)	Set track output Current Limit. (milliamps) (adjustable in 50mA increments)

Configuration Parameters:

PARAMETER NAME	VALUE (range/options)	DESCRIPTION
Code5	**Standard**, **Long**, or **Alternating**	Select Code 5 format.
EC4Compatibility	**EC4**, or **EC4Plus**	Select EC4 Compatibility option.
NonvitalCodeChangeCycles	**1 – 3 cycles**	Number of code cycles before changing from one non-vital code to another non-vital code.
VitalCodeChangeCycles	**1 – 3 cycles**	Number of code cycles: Before changing from one vital code to another vital code. Before changing from a non-vital code to a vital code. Before changing from a vital code to a non-vital code.
ShuntDropCycles	**1 – 8 cycles**	Number of code cycles that a code is reported after a shunt is applied.
ShuntPickCycles	**1 – 8 cycles**	Number of code cycles required after a shunt is picked before a code 1 is reported.

2.9.1.3 Coded Line

Output channels: LED 1
Bidirectional channel: CTIO 1

Operating Parameters:

PARAMETER NAME	VALUE (range/options)	DESCRIPTION
Transmit Voltage	**0** thru **4000 mV** (by 20's)	Set coded line transmit voltage. (millivolts)
Receive Threshold	**1000** thru **10000 mA** (by 50's)	Set line output Current limit. (milliamps)

Configuration Parameters:

PARAMETER NAME	VALUE (range/options)	DESCRIPTION
Code5	**Standard, Long**, or **Alternating**	Select Code 5 format.
EC4Compatibility	**EC4**, or **EC4Plus**	Select EC4 Compatibility option.
NonvitalCodeChangeCycles	**1 – 3 cycles**	Number of code cycles before changing from one non-vital code to another non-vital code.
VitalCodeChangeCycles	**1 – 3 cycles**	Number of code cycles: Before changing from one vital code to another vital code. Before changing from a non-vital code to a vital code. Before changing from a vital code to a non-vital code.
ShuntDropCycles	**1 – 8 cycles**	Number of code cycles that a code is reported after a shunt is applied.
ShuntPickCycles	**1 – 8 cycles**	Number of code cycles required after a shunt is picked before a code 1 is reported.

2.9.1.4 RIO

Output channels: VRO 1-4
Input channels: VPI 1-4

Operating Parameters:

PARAMETER NAME	VALUE (range/options)	DESCRIPTION
VPIDebounce	**20** thru **200 ms** (by 2's)	Set VPI Debounce time. (milliseconds) (adjustable in 2ms increments)

Configuration Parameters:
None

2.9.1.5 VPI

Output channels: none
Input channels: VPI 1-8

Operating Parameters:

PARAMETER NAME	VALUE (range/options)	DESCRIPTION
VPIDebounce	**20** thru **200 ms** (by 2's)	Set VPI Debounce time. (milliseconds) (adjustable in 2ms increments)

Configuration Parameters:
None

2.9.1.6 VRO

Output channels: VRO 1-6
Input channels: None

Operating Parameters:
None

Configuration Parameters:
None

2.9.1.7 Colorlight

Output channels: VRO 1
Input channels: VPI 1-2
Bidirectional channels: VLO 1-6

Operating Parameters:

PARAMETER NAME	VALUE (range/options)	DESCRIPTION
LampVoltage	**9000** thru **13000 mV** (by 20's)	Set Lamp Voltage. (millivolts) (adjustable in 20 mV increments)
LampFilamentThreshold	**150** thru **2500 mA** (by 10's)	Sets lamp filament threshold to determine light out detection.
VPIDebounce	**20** thru **200 ms** (by 2's)	Set VPI Debounce time. (milliseconds) (adjustable in 2ms increments)
LampVoltageReg	Constant or Variable	Set to Constant to maintain constant lamp output voltage even under low battery conditions.

Configuration Parameters:

PARAMETER NAME	VALUE (range/options)	DESCRIPTION
PerformColdFilamentTest	**BYTEYESNOTYPE_YES, BYTEYESNOTYPE_NO**	Enable / disable cold filament test.
VLOFlashRate	**VLOFLASHRATE_40cpm, VLOFLASHRATE_45cpm, VLOFLASHRATE_50cpm, VLOFLASHRATE_55cpm, VLOFLASHRATE_60cpm, VLOFLASHRATE_65cpm**	Lamp flash rate in cycles per minute (CPM) with 50% duty cycle.
RedRetainingRelayEnabled	**RRTYPE_NORELAY RRTYPE_ONTHISMODULE RRTYPE_ ONANOTHERMODULE**	Enable / disable red retaining relay.
RedRetainingGroup1	**CLSSLOTTYPE_NORELAY, CLSSLOTTYPE_SLOT2, CLSSLOTTYPE_SLOT3, CLSSLOTTYPE_SLOT4, CLSSLOTTYPE_SLOT5, CLSSLOTTYPE_SLOT6, CLSSLOTTYPE_SLOT7, CLSSLOTTYPE_SLOT8**	Specifies a group of colorlight cards (by slot number) for red retaining relay management.

PARAMETER NAME	VALUE (range/options)	DESCRIPTION
RedRetainingGroup2	**CLSSLOTTYPE_NORELAY, CLSSLOTTYPE_SLOT2, CLSSLOTTYPE_SLOT3, CLSSLOTTYPE_SLOT4, CLSSLOTTYPE_SLOT5, CLSSLOTTYPE_SLOT6, CLSSLOTTYPE_SLOT7, CLSSLOTTYPE_SLOT8**	Specifies a group of colorlight cards (by slot number) for red retaining relay management.
RedRetainingGroup3	**CLSSLOTTYPE_NORELAY, CLSSLOTTYPE_SLOT2, CLSSLOTTYPE_SLOT3, CLSSLOTTYPE_SLOT4, CLSSLOTTYPE_SLOT5, CLSSLOTTYPE_SLOT6, CLSSLOTTYPE_SLOT7, CLSSLOTTYPE_SLOT8**	Specifies a group of colorlight cards (by slot number) for red retaining relay management.

2.9.1.8 VLP2

Output channels: LED 1-16
Input channels: None

Operating Parameters:

PARAMETER NAME	VALUE (range/options)	DESCRIPTION
DaylightSavings	**BYTEONOFFTYPE_OFF, BYTEONOFFTYPE_ON**	Enable / disable daylight savings function.
Units	**Standard** or **Metric**	Not currently supported.
LowBatteryThresholdEnabled	**BYTEONOFFTYPE_OFF, BYTEONOFFTYPE_ON**	Enable / disable low battery detection.
LowBatteryThreshold	**90 – 150 dv** (by 1)	Set low battery detection level. (adjustable in 1 decivolt increments)
RadioSubnode	**1 - 99**	Set radio subnode.
PasswordActive	**BYTEONOFFTYPE_OFF, BYTEONOFFTYPE_ON**	Enable / disable password function.
Password4	**1111 – 9999**	Set 4-digit password.

Configuration Parameters:

None

2.9.2 I/O Configuration

2.9.2.1 Relays and Coded Track

io_config

ASSIGN (*object, input* **) (slot** *constant* **, VPI** *constant* **)**
ASSIGN (*object, output* **) (slot** *constant* **, VRO** *constant* **)**
ASSIGN (*object, coded-track-io* **) (slot** *constant* **, CTIO** *constant* **)**

object is an identifier that specifies a geographic object from the track layout.

input is an identifier that specifies an input of *object* (input type must be Boolean).

The same VPI channel inputs may be assigned to different object inputs; the same object output may be assigned to different VRO channel outputs.

2.9.2.2 Colorlight

io_config

ASSIGN (*object, aspect-io* **) (**
lamp **(slot** *constant* **, VLO** *constant* **),** { *lamp* **(slot** *constant* **, VLO** *constant* **),** …
}
)

object is an identifier that specifies a geographic object from the track layout.

lamp is a two character identifier that specifies a colorlight signal lamp: the first character specifies a signal head (**A**, **B**, or **C**) and the second character specifies a lamp (**G**, **Y**, **R**, or **L**). For example, AG for head A, green lamp; or BL for head B, lunar lamp.

2.9.2.3 Searchlight

io_config

ASSIGN (*object, aspect-io* **) (**
head **(slot** *constant* **, VLO** *constant* **),** { *lamp* **(slot** *constant* **, VLO** *constant* **),** …
}
)

object is an identifier that specifies a geographic object from the track layout.

head is a single character identifier that specifies a searchlight signal head (**A**, **B**, or **C**). Pole change outputs and mechanism detection inputs corresponding to select VLO channel on the searchlight will automatically be assigned to control the specified signal head.

2.9.2.4 Internal Channel

io_config

ASSIGN (*object, input-output* **) (INTCH** *constant* **)**

object is an identifier that specifies a geographic object from the track layout.

2.9.3 Controls

Use a control statement to assign controls to specific control bits.

CONTROL

CONTROL *name { ***[***description-qualifier***]*** } ***(*** control-bit {, control-bit, …} ***)**

name may be any *identifier* used to identify the control message.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the control message.

control-bit

(*object* , *input* **) [bit =** constant **]**

control-bit assigns an object input to a specific control bit position.

object is an identifier that specifies a geographic object from the track layout.

input is an identifier that specifies an input of *object* (input type must be Boolean).

bit = constant

The same control bit may be assigned to different object inputs.

2.9.4 Indications

Use an indication statement to assign indications to specific indication bits.

INDICATION

INDICIATION *name { ***[***description-qualifier***]*** } ***(*** indication -bit {, indication-bit, …} ***)**

name may be any *identifier* use to identify the indication message.

description-qualifier

desc = *string*

description-qualifier specifies an optional text string describing the indication message.

indication-bit

(*object* , *output* **) [bit =** constant **]**

indication-bit assigns an object output to a specific indication bit position.

object is an identifier that specifies a geographic object from the track layout.

output is an identifier that specifies an output of *object* (output type must be Boolean).

bit = constant

The same object output may be assigned to different indication inputs.

SECTION 3
GEO SCRIPTING

3.0 GEO SCRIPTING

3.1 THE ELEMENTS SCRIPTING

The functionality of a geographic interlocking is defined by the following components: the specific interconnection of a set of Geographic Objects (i.e. the track layout); the logic associated with each type of Geographic Object referenced in the track layout (i.e. the logic defined in the Geographic Object Library); the aspect sequences assigned to the routes in the track layout; the settings of the configurable properties of each Geographic Object.

The functionality (i.e. logic) of a geographic interlocking as defined by these components may be tested by performing interactive simulations using the GEO Logic Simulator. As discussed previously, interactive simulation of a track layout involves exercising the inputs of Geographic Objects and verifying expected outputs of Geographic Objects by observing the states of the outputs on either the track layout or on the I/O View tab in the Workspace. GEO Scripting is a mechanism for formalizing interactive simulations of a geographic interlocking.

3.1.1 Keywords for Writing GEO Script (.gs) Files

The following words are reserved for use as keywords in GEO Script (.gs) files, and may not be used otherwise. Note that keywords are case-sensitive in that lowercase keywords are "qualifiers".

ASPECT	**aspect**	**AUTO**
CABOUT	**CLOSE**	**CODEIN**
CODEIN2	**CODEOUT**	**DEENERGIZED**
DG	**ENERGIZED**	**EXPIRE**
FOREIGNENERGY or FE	**GROUP**	**HG**
HIGH	**IGNORE**	**LOG**
LOW	**MANUAL**	**MECHFAIL**
MECHPOS	**mm**	**NC**
NOCODE	**OPEN**	**OPTION**
PROPERTY	**QUICKPICK**	**QUICKSHUNT**
RESET	**RG**	**route**
SELECT	**SET**	**ss**
TIMER	**VERIFY**	**WAIT**

3.1.2 Keywords for Writing GEO Script Project (.gsp) Files

The following words are reserved for use as keywords in GEO Script Project (.gsp) files, and may not be used otherwise. Note that keywords are case-sensitive.

COMMON	**DEFINE**	**DO**
LOCAL	**MAP**	**PROJECT**
RESET		

3.1.3 GEO Scripts

A GEO Script is a text file with a .gs extension created using any ASCII text editor (E.g. Windows Notepad).

A GEO Script is a series of statements that defines a sequence of Geographic Object inputs followed by statements that specify expected Geographic Object outputs. Statements that set Geographic Object inputs are called input statements. Geographic Object inputs include CTC controls, physical inputs, field configurable inputs, and expiration of timers. Statements that specify expected Geographic Object outputs (i.e. statements that *verify* Geographic Object outputs) are called output statements. Geographic Object outputs include CTC indications, physical outputs, and the starting of timers.

For example, consider an intermediate signal location (Figure 3-1 and the following sample GEO Script (Approach.gs) that simulates a westbound train approaching the intermediate signal location.

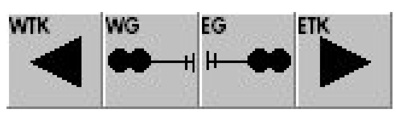


Figure 3-1 Example Track Layout

Approach.gs

```
SET ETK QUICKSHUNT        ; put east track into quick shunt mode
CODEIN ETK NOCODE         ; set no code received from the east
ASPECT EG R/R             ; verify east signal aspect is R/R
CODEOUT WTK C1,C6         ; C1, C6 is transmitted on west track
CODEOUT ETK C1,C8         ; C1, C8 is transmitted on east track
```

Referring to the Approach.gs script, the first line of the script is an input statement that puts the east Coded Track object (ETK) into quick shunt mode – this is the first sign of a train approaching from the east. The second line of the script is an input statement indicating no code is received from the east (i.e. east track is occupied).

The last three lines in the script are statements that verify the expected outputs of the Geographic Objects in the track layout – expected outputs that are a result of the occupancy on the east track. Specifically, the third line of the script verifies that the east signal is displaying R/R, the fourth line of the script verifies that C1 and C6 are transmitted on the west track, and the fifth line of the script verifies that C1 and C8 are transmitted on the east track.

The expected outputs specified in the sample script above depend on the condition (i.e. the state of the inputs/outputs) of the track layout prior to the execution of the script. Since the sample script does not specify the state of the track layout prior to the execution of the script, it is not considered to be complete. The sample script may be completed by inserting necessary input/output statements at the beginning of the script to set up the initial conditions expected by the rest of the script. This issue will be explained in the following sections. At this time it is enough to note that it is important to consider the state of the track layout prior to the execution of a script.

3.1.4 GEO Script Execution Sequence

Refer to the intermediate signal location from the example in the previous section and consider two GEO scripts: a complete westbound train movement across the intermediate signal location that *does not* trigger the westbound joint-hop logic (TrainMove.gs), and a complete westbound train movement across the intermediate signal location that *does* trigger the westbound joint-hop logic (TrainMoveJointHop.gs) as shown in the following listings.

TrainMove.gs

```
; Approach
VERIFY ETK.STICK False     ;
SET ETK QUICKSHUNT         ;
CODEIN ETK NOCODE          ;
ASPECT EG R/R              ;
CODEOUT WTK C1,C6          ;
CODEOUT ETK C1,C8          ;

; Move
SET WTK QUICKSHUNT         ;
ASPECT WG R/R              ;
CODEOUT ETK C1,C8          ;
VERIFY ETK.STICK True      ;
CODEIN WTK NoCode          ;

; Unoccupy Approach
SET ETK QUICKPICK          ;
CODEIN ETK C1              ;

; Complete
SET WTK QUICKPICK          ;
CODEIN WTK C1              ;
VERIFY ETK.STICK True      ;
CODEIN WTK C1,C2           ;
VERIFY ETK.STICK False     ;
ASPECT WG Y/R              ;
CODEOUT ETK C1,C3          ;
```

TrainMoveJointHop.gs

```
; Approach
VERIFY ETK.STICK False     ;
SET ETK QUICKSHUNT         ;
CODEIN ETK NOCODE          ;
ASPECT EG R/R              ;
CODEOUT WTK C1,C6          ;
CODEOUT ETK C1,C8          ;

; JointHop
SET ETK QUICKPICK          ;

; Move
SET WTK QUICKSHUNT         ;
ASPECT WG R/R              ;
CODEOUT ETK C1,C8          ;
VERIFY ETK.STICK True      ;
CODEIN WTK NoCode          ;

; Unoccupy Approach
CODEIN ETK C1              ;
```

Both scripts begin by simulating a train approaching the intermediate signal location from the east (just as the Approach script did in the example from the previous section). In both scripts, the remaining input/output statements expect that the condition (i.e. the state of the inputs/outputs) of the layout to be such that a train is approaching from the east. There might be many scenarios that initially require the track layout to be set up for a train approaching from the east. GEO Scripts for all these scenarios would duplicate the same set of input/output statements to simulate an approaching train.

To avoid duplication and minimize the effect of errors, common scenarios such as an approaching train move may be simulated in separate GEO Scripts. For example, the TrainMove and TrainMoveJointHop scripts may be divided into three scripts to avoid duplication of the Approach scenario, as shown below.

Approach.gs

```
; Approach
VERIFY ETK.STICK False     ;
SET ETK QUICKSHUNT         ;
CODEIN ETK NOCODE          ;
ASPECT EG R/R              ;
CODEOUT WTK C1,C6          ;
CODEOUT ETK C1,C8          ;
```

TrainMove2.gs

```
; Move
SET WTK QUICKSHUNT         ;
ASPECT WG R/R              ;
CODEOUT ETK C1,C8          ;
VERIFY ETK.STICK True      ;
CODEIN WTK NoCode          ;

; Unoccupy Approach
SET ETK QUICKPICK          ;
CODEIN ETK C1              ;

; Complete
SET WTK QUICKPICK          ;
CODEIN WTK C1              ;
VERIFY ETK.STICK True      ;
CODEIN WTK C1,C2           ;
VERIFY ETK.STICK False     ;
ASPECT WG Y/R              ;
CODEOUT ETK C1,C3          ;
```

TrainMoveJointHop2.gs

```
; JointHop
SET ETK QUICKPICK          ;

; Move
SET WTK QUICKSHUNT         ;
ASPECT WG R/R              ;
CODEOUT ETK C1,C8          ;
VERIFY ETK.STICK True      ;
CODEIN WTK NoCode          ;
```

The creation of Approach, TrainMove2 and TrainMoveJointHop2 implies two GEO Script execution sequences: 1) execute Approach; execute TrainMove2, and 2) execute Approach; execute TrainMoveJointHop2. The first sequence performs the simulation specified by the original TrainMove script, and the second sequence performs the simulation specified by the original TrainMoveJointHop script. These execution sequences are illustrated in Figure 3-2.

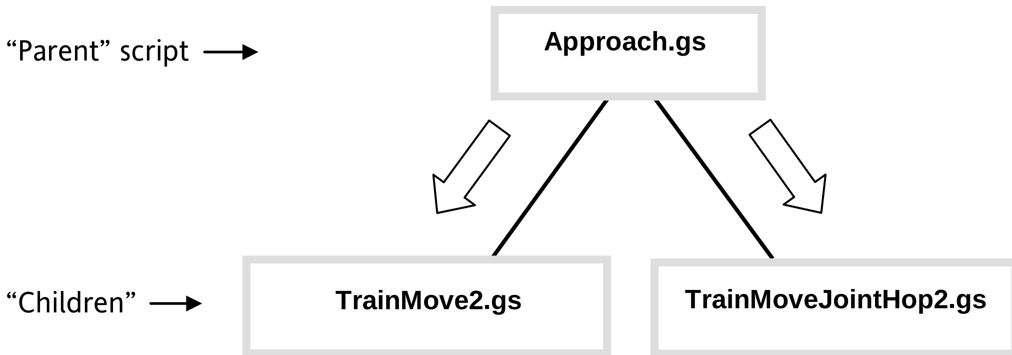


Figure 3-2. Two Script Execution Sequences

The execution sequence for the above illustration is:

APPROACH, TRAINMOVE2
APPROACH, TRAINMOVEJOINTHOP2

Expanding on the concept of creating scripts to perform repetitious tasks, we'll add a scenario that follows the JointHop script where shunt pick time has expired, and add it to the execution sequence. The listing –Expired.gs–is shown below and the sequence is illustrated in Figure 3-3.

Expired.gs

```
CODEIN ETK C1              ;
CODEOUT ETK C1,C2          ;

; Move (failed to set stick)
SET WTK QUICKSHUNT         ;
VERIFY ETK.STICK False     ;
CODEIN WTK NoCode          ;
VERIFY ETK.STICK False     ;
ASPECT WG R/R              ;
CODEOUT ETK C1,C6          ;
```

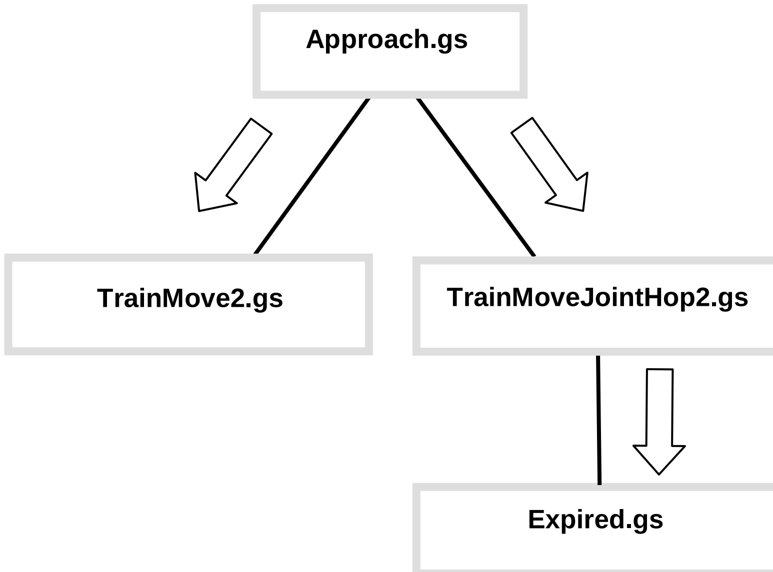


Figure 3-3 Three Script Execution Sequences

The execution sequence for the above illustration is:

APPROACH, TRAINMOVE2
APPROACH, TRAINMOVEJOINTHOP2, EXPIRED

When a GEO Logic Script is executed, the track layout is left in its simulated state after the script has completed. If multiple scripts are executed in sequence on the same track layout, each succeeding script begins at the state where the preceding script ended. This is called an execution sequence.

As can be seen from the previous examples, the scripts exercise some of the functions of the Geographic Objects, but do not take into account the previous state or conditions the layout was in prior to their execution. Since a GEO Logic Script will, in most cases, transform the track layout from one state to another, it would be beneficial to precede the Approach script with a script to “initialize” the layout, or set it to some predefined state before executing the Approach sequence.

The following listing—Setup.gs—is an example of an initialization script which predefines the condition of the Geographic Objects to ensure the state of the layout prior to the execution of the Approach script (and others). The sequence is illustrated in Figure 3-4.

Setup.gs

```
CODEOUT ETK C1,C6          ;

GROUP                      ;
{                          ;
CLOSE WG                   ;
CLOSE EG                   ;

SET WTK.AUX True           ;
SET ETK.AUX True           ;

SET WG.ACP True            ;

CODEIN WTK C1,C2           ;
CODEIN ETK C1,C2           ;
}                          ;

ASPECT WG Y/R              ;
CODEOUT ETK C1,C4          ;
ASPECT EG Y/R              ;
CODEOUT WTK C1,C4          ;
```

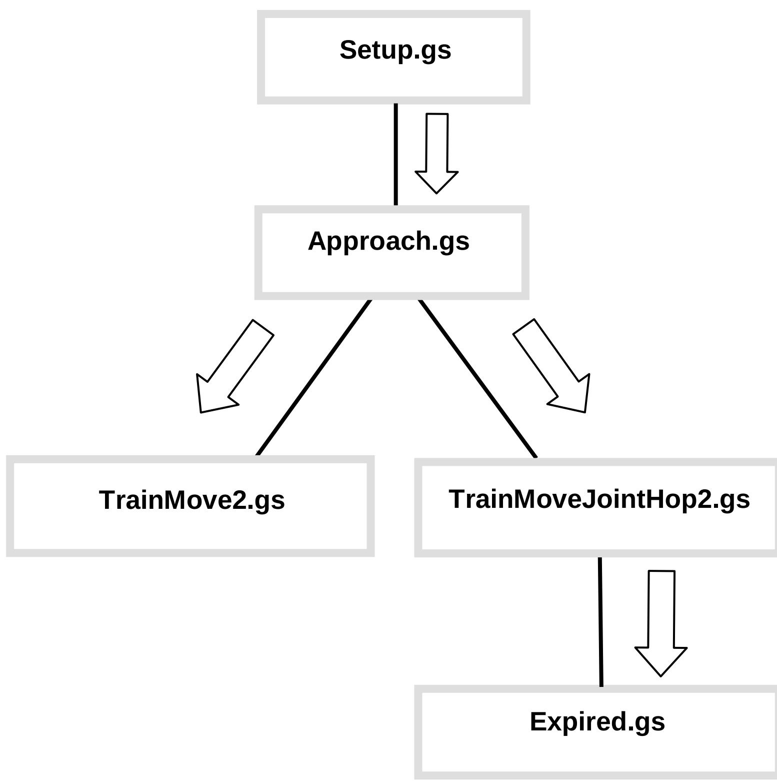


Figure 3-4 Adding an Initialization Script (Setup.gs)

3.1.5 The GEO Script Project File

The GEO Script Project File provides a means to organize a set of individual GEO Script files into execution sequences.

GEO Script Project Files are ASCII text files with a "**.gsp**" filename extension. They can be created and edited with a simple text editor such as Windows **NotePad**.

The following is an example of a Project File.

```
; LFCINT.GSP                            ; Project File located in root directory

PROJECT Main                            ; Main Project routine
{                                       ; Beginning of 1st script (parent)
     Setup                              ; Setup.gs also located in root directory
     {                                  ; Beginning of child script
            Approach                    ; Approach.gs located in Setup subdirectory
            {                           ; Beginning of child script
                   JointHop             ; JointHop.gs located in root\Setup\Approach
                   {                    ; Beginning of child scripts
                          Move          ; Root\Setup\Approach\JointHop\Move.gs
                          Expired       ; Root\Setup\Approach\JointHop\Expired.gs
                   }                    ; End of child scripts
                   Move                 ; Move.gs located in root\Setup\Approach
                   {                    ; Beginning of child scripts
                          Complete      ; Root\Setup\Approach\Move\Complete.gs
                          AuxIn         ; Root\Setup\Approach\Move\AuxIn.gs
                   }                    ; End of child scripts
            }                           ; End of child scripts
     }                                  ; End of child scripts
}                                       ; End of Main routine
```

A GEO Script Project File is a collection of **PROJECT** statements. Each **PROJECT** statement defines a partial execution sequence by specifying a hierarchy of GEO Script files. The execution sequence begins with a **PROJECT** called Main. A GEO Script Project File must always contain a **PROJECT** statement with the name MAIN.

This Page Intentionally Left Blank

SECTION 4
TUTORIAL – GC FILE CREATION

4.0 TUTORIAL – GC FILE CREATION

4.1 SECTION SUMMARY

This section is a tutorial on using the GEO Configuration Suite (GCS) to develop a typical intermediate signal application.

4.2 TYPICAL INTERMEDIATE SPECIFICATIONS

The tutorial is based on an intermediate signal application using the aspects and codes provided in Table 4-1.

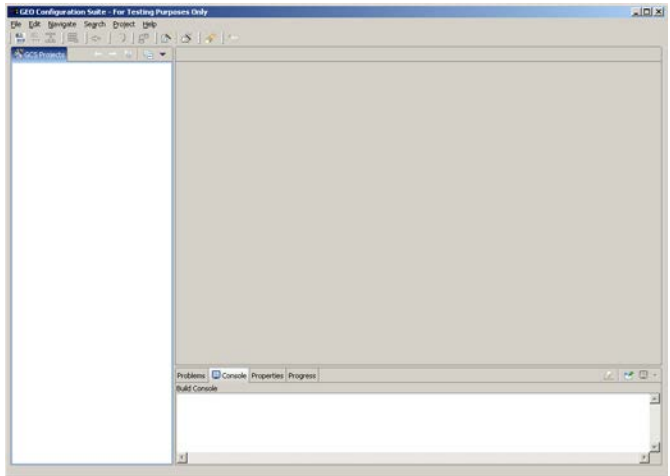
Table 4-1 Aspect/Code Chart for Typical Intermediate Signal Application

West Code In	West Signal Aspects	East Code Out	East Code In	East Signal Aspects	West Code Out
7	G/R blo G/D tlo FY/R	7 7 7	7	G/R blo G/D tlo FY/R	7 7 7
4	FY/R blo FY/D tlo D/R	7 7 2	4	FY/R blo FY/D tlo D/R	7 7 2
3	Y/Y blo Y/D tlo D/Y	7 7 7	3	Y/Y blo Y/D tlo D/Y	7 7 7
2	Y/R blo Y/D tlo D/R	4 4 2	2	Y/R blo Y/D tlo D/R	4 4 2
8	Y/L blo Y/D tlo D/L	4 4 2	8	Y/L blo Y/D tlo D/L	4 4 2
1	R/R	1	1	R/R	1
No Code	R/R	1	No Code	R/R	1
West Stick Code 2			East Stick Code 2		

G = Green, Y = Yellow, R = Red, FY = Flashing Yellow, L = Lunar, D = Dark, blo = bottom lamp out, tlo = top lamp out

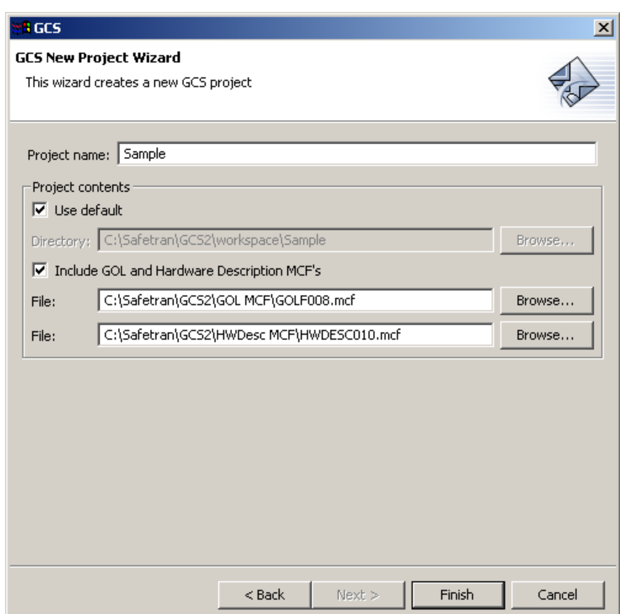
4.3 START THE GCS

The GCS contains three panels: the left panel is called the **Project View**, the top right panel is called the **Editor View**, and the bottom right panel is called the **Status View**.



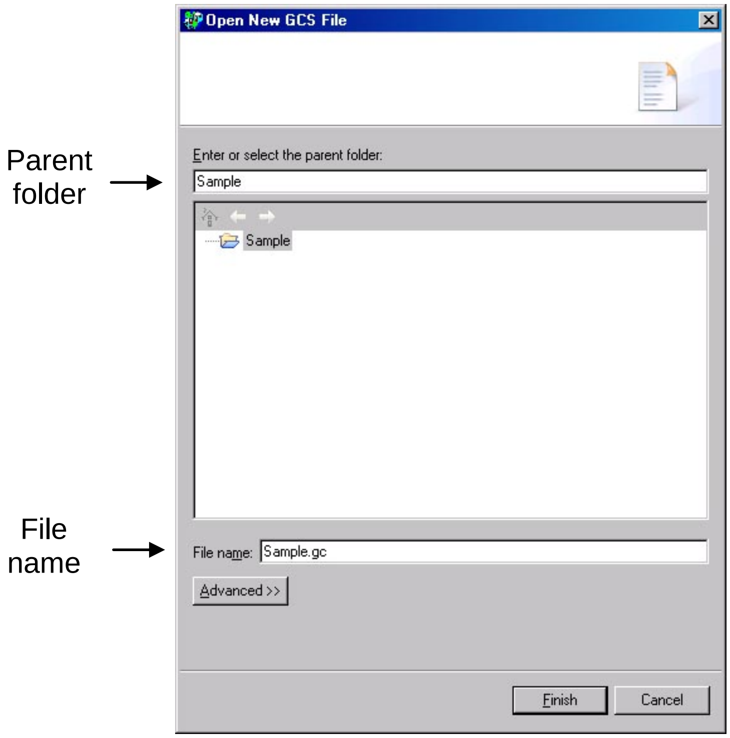
4.4 CREATE NEW GCS PROJECT

1. Click **File**
2. Select **New** > **Project**
3. Enter **Sample** as the **Project name**. For the project contents directory leave **Use default** checked, or uncheck the box and browse to select a different project directory. Ensure **GOL and Hardware Description MCF's** is checked, then click **Finish**.



4.5 CREATE NEW GEO CONFIGURATION (GC) FILE

1. Click **File**
2. Select **New > File**
3. Select **Sample** as the parent folder.
4. Type **Sample.gc** as file name and click **Finish**. The Sample.gc file is created and opened in the **Editor View**.



4.6 CREATE ASPECT CHART

Create Aspect Chart named **Sample** by typing the following in the **Sample.gc** file:

```
ASPECTCHART Sample [desc="Aspect Definitions"]
(
      ASPECTDEF Clear (G/R, G/D),
      ASPECTDEF "Approach Limited" (FY/R, FY/D),
      ASPECTDEF "Approach Medium" (Y/Y),
      ASPECTDEF Approach (Y/R, Y/D),
      ASPECTDEF "Approach Restricting" (Y/L),
      ASPECTDEF Restricting (D/L),
      ASPECTDEF "Stop And Proceed" (R/R),
      ASPECTDEF CODE7,
      ASPECTDEF CODE4,
      ASPECTDEF CODE3,
      ASPECTDEF CODE2,
      ASPECTDEF CODE8,
      ASPECTDEF CODE9
)
```

4.7 CREATE ASPECT SET

Create Aspect Set named **2_Head_Automatic** by typing the following in the **Sample.gc** file:

```
ASPECTSET 2_Head_Automatic [desc=""]
(
      G/R, G/D,    # Clear
      FY/R, FY/D, # Approach Limited
      Y/Y,         # Approach Medium
      Y/R, Y/D,    # Approach
      Y/L,         # Approach Restricting
      D/L,         # Restricting
      R/R [aspect="Stop And Proceed", stopaspect]
)
```

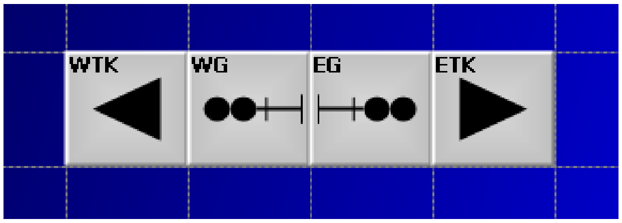
4.8 CREATE DEGRADED ASPECT TABLE

Create Degraded Aspect Table for the **2_Head_Automatic** Aspect Set by typing the following in the **Sample.gc** file:

```
DEGASPECTTABLE [aspectset=2_Head_Automatic]
(
      (G/R, G/D),        # Clear to Clear
      (G/D, FY/R),       # Clear to Approach Limited
      (FY/R, FY/D),      # Approach Limited to Approach Limited
      (FY/D, R/R),       # Approach Limited to Stop And Proceed
      (Y/Y, Y/R),        # Approach Medium to Approach
      (Y/R, Y/D),        # Approach to Approach
      (Y/D, R/R),        # Approach to Stop And Proceed
      (Y/L, Y/D),        # Approach Restricting to Approach
      (D/L, R/R),        # Restricting to Stop And Proceed
      (R/R, R/R)         # Stop And Proceed to Stop And Proceed
)
```

4.9 CREATE TRACK LAYOUT

In this step, Geographic Objects will be used to lay out the typical intermediate signal application as shown here.



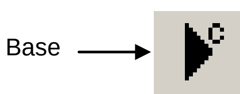
1. Create Track Layout section named **LFCINT** by typing the following in the **Sample.gc** file:

```
   TRACKLAYOUT LFCINT [desc=""]
   (
)
```

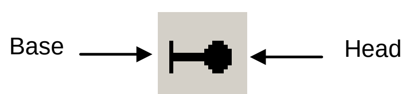
1. Click on **Sample.gc** in the Project View to select it.
2. Click **Edit**
3. Select **Edit Track Layout**
4. The empty **Sample.gc** track layout opens in the Editor View.

Menu Button

5. Click the **Menu button** on the Geographic Object Toolbar to display a drop-down menu with icons and short text descriptions of each button on the toolbar. Alternatively, hovering the cursor over an icon on the toolbar will display a "Tool Tip" with the name of the tool or its function. Selecting an item in the drop-down menu or on the toolbar performs the same function.
6. Click to select the **Coded Track** object (from either the drop-down menu or the toolbar).
7. Move the cursor arrow slightly to the left-of-center in the track layout window and click to place the **Coded Track** object.
8. Click to select the **Signal** object on the toolbar or the drop-down menu.
9. Move the cursor arrow to the track layout window and click to place the **Signal** object in the grid box to the right of the **Coded Track** object.
10. With the **Signal** object still highlighted, click the **Flip Horizontal** button on the Toolbar (or the **Rotate** button twice) so the signal object's head is next to the base of the coded track object. (Refer to the sample track layout below).

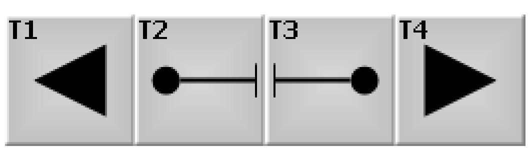


Coded Track Object



Signal Object

11. Select and place a second **Signal** object in the grid box to the right of the first Signal object.
12. Select and place a second **Coded Track** object to the right of the second Signal object.
13. With the second **Coded Track** object still highlighted, click the **Flip Horizontal** button (or the **Rotate** button twice) so the base is next to the head of the second Signal object.
14. The track layout should now be assembled as shown:

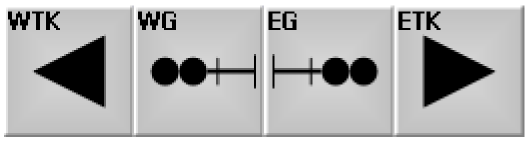


Sample Track Layout

4.10 NAME OBJECTS

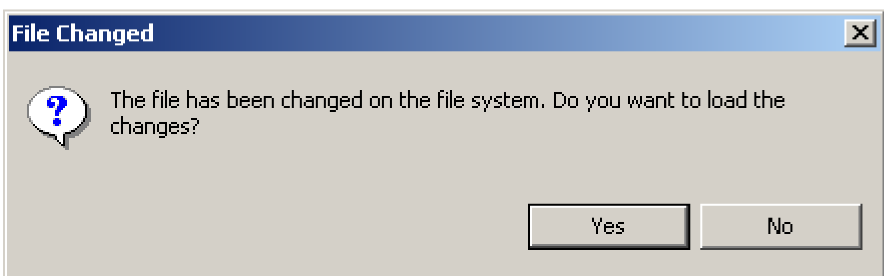
Quick Reference: Select object > Right-click object > Rename

1. Right-click on the left Coded Track object and select **Rename** from the popup menu. The Rename SAT Object dialog box appears.
2. Click in the Name field and type **WTK**. Click **OK** or press **ENTER**.
3. Repeat steps 1 & 2 to rename the left Signal object **WG**.
4. Repeat steps 1 & 2 to rename the right Signal object **EG**.
5. Repeat steps 1 & 2 to rename the right Coded Track object **ETK**.



4.11 SAVE TRACK LAYOUT

1. Click the **Save** button on the toolbar of the track layout. 
2. Close the track layout by clicking the **X** on the Sample.gc tab.
3. When the **File Changed** dialog box appears, click **YES** to load the changes.



4.12 ACTIVATE I/O

Activate the I/O by typing the following statements into the TRACKLAYOUT section in the **Sample.gc** file. Insert the ACTIVATE statements after the LINK statements in the TRACKLAYOUT section, and before the closing parenthesis. You will need to scroll up through the Sample.gc file to locate the TRACKLAYOUT section, since it was created in an earlier step.

```
ACTIVATE WTK (CT, AUX, HR)
ACTIVATE ETK (CT, AUX, HR)
ACTIVATE WG (sig)
ACTIVATE EG (sig)
```

4.13 DEFINE TABLESET SECTION

Create TABLESET section by typing the following in the **Sample.gc** file after the TRACKLAYOUT section:

```
TABLESET tsSAMPLE [desc=""]
(
# Insert Tables Here
)
```

4.14 CREATE TABLES

The tables below should be inserted between the parentheses of the TABLESET section that was created in the previous step.

1. Create Code In Table named **CIT** by typing the following in the TABLESET section of the **Sample.gc** file:

```
TABLE CIT [desc="Code In Table"]
(
      Codes [enum=Codes, context=VitalCodes, desc="Code In"],
      Aspects [ASPECT, desc="Aspect"]) = (
(C7, CODE7),
            (C4, CODE4),
            (C3, CODE3),
            (C2, CODE2),
            (C8, CODE8),
            (C9, CODE9)
)
```

2. Create Normal Aspect Table named **LFCINT_NAT** by typing the following in the TABLESET section of the **Sample.gc** file:

```
TABLE LFCINT_NAT [desc="Normal Aspect Table"]
(
      Left [ASPECT, desc="Exit Aspect"],
      Right [ASPECT, context=2_Head_Automatic, desc="Home Aspect"]) = (
            (CODE7, G/R),
            (CODE4, FY/R),
            (CODE3, Y/Y),
            (CODE2, Y/R),
            (CODE8, Y/L),
            (CODE9, R/R)
)
```

3. Create Code Output Table named **LFCINT_COT** by typing the following in the TABLESET section of the **Sample.gc** file:

```
TABLE LFCINT_COT [desc="Codes Out"]
(
      Aspects [ASPECT, desc="Aspect"],

      Codes [enum=Codes, context=VitalCodes, desc="Code Out"]) = (
            (Clear, C7),
            ("Approach Limited", C7),
            ("Approach Medium", C7),
            (Approach, C4),
            ("Approach Restricting", C4),
            (Restricting, C2),
            ("Stop And Proceed", C2)
)
```

4.15 DEFINE LOGICALCONFIG SECTION

Create LOGICALCONFIG section named **lcLFCINT** by typing the following in the **Sample.gc** file after the last closed parenthesis of the TABLESET section:

```
LOGICALCONFIG lcLFCINT [desc=""]
(
      OBJECT WTK
      (
            CodeIn := CIT
            OutAspect := LFCINT_COT
      )

      OBJECT WG
      (
            AspectSet = 2_Head_Automatic
            SigType = SIGTYPE_AUTO
            NorAspect := LFCINT_NAT
      )

      OBJECT EG
      (
            AspectSet = 2_Head_Automatic
            SigType = SIGTYPE_AUTO
            NorAspect := LFCINT_NAT
      )

      OBJECT ETK
      (
            CodeIn := CIT
            OutAspect := LFCINT_COT
      )
)
```

Alternatively, the logical configuration section could be created by utilizing the **lc.tmpl.gc** file as follows:

1. Save changes by clicking File and selecting Save.
2. An lc.tmpl.gc logical configuration template file was added to the project after the track layout was created, and is visible in the GCS Projects view. Open this file by double-clicking it.
3. The properties are set to their default values in the template, and all other possible values for each property are included as comments.
4. Copy the contents of the lc.tmpl.gc file and paste into the Sample.gc file following the TABLESET section.
5. Change the property values and assign the tables as needed.

4.16 DEFINE ATCSCONFIG SECTION

Create ATCSCONFIG section named **LFCINT_RA** by typing the following in the **Sample.gc** file after the LOGICALCONFIG section:

```
ATCSCONFIG atcsLFCINT
(
      RAILROAD LFCINT_RA [number=620] (
            LINE LFCINT_LI [number=120] (
                  GROUP LFCINT_GR [number=100] (
                        SUBNODE LFCINT_SU [number=3] (
                              DEVICE WTK
                              DEVICE WG
                              DEVICE EG
                              DEVICE ETK
                        )
                  )
            )
      )
)
```

4.17 DEFINE GWECONFIG SECTION

Create a GWECONFIG section named **gweLFCINT** by typing the following in the **Sample.gc** file after the ATCSCONFIG section:

NOTE

NOTE

If, after adding the following text, you perform a file save before activating the I/O (as described in the next step) you may receive an I/O error message after the compilation. This is normal.

```
GWECONFIG gweLFCINT [atcsconfig=atcsLFCINT, subnode=LFCINT_SU,
gwe=TwoTrackExtended]
(
      MODULE sl1 [slot=1, module=VLP2]
      MODULE sl2 [slot=2, module=CodedTrack]
      MODULE sl3 [slot=3, module=Colorlight]
      MODULE sl5 [slot=5, module=Colorlight]
      MODULE sl8 [slot=8, module=CodedTrack]

      ASSIGN (WTK, CT) (slot 2, CTIO 1)
      ASSIGN (WTK, AUX) (slot 3, VPI 1)
      ASSIGN (ETK, AUX) (slot 5, VPI 1)
      ASSIGN (WTK, HR) (slot 3, VRO 1)
      ASSIGN (ETK, HR) (slot 5, VRO 1)
      ASSIGN (ETK, CT) (slot 8, CTIO 1)

      ASSIGN (EG, sig) (
            AG (slot 3, VLO 1),
            AY (slot 3, VLO 2),
            AR (slot 3, VLO 3),
            BL (slot 3, VLO 4),
            BY (slot 3, VLO 5),
            BR (slot 3, VLO 6)
      )

      ASSIGN (WG, sig) (
            AG (slot 5, VLO 1),
            AY (slot 5, VLO 2),
            AR (slot 5, VLO 3),
            BL (slot 5, VLO 4),
            BY (slot 5, VLO 5),
            BR (slot 5, VLO 6)
      )
)
```

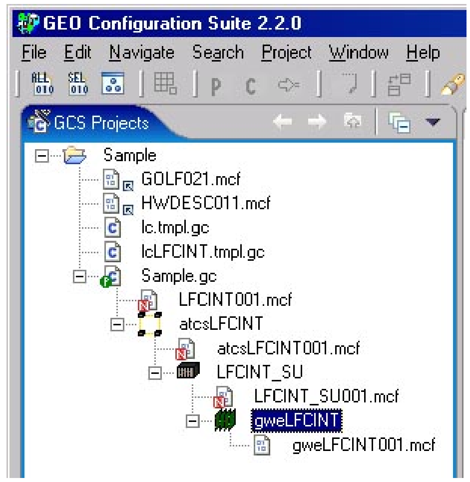
4.18 SYNTAX CHECK

The syntax checker automatically executes each time changes are saved.

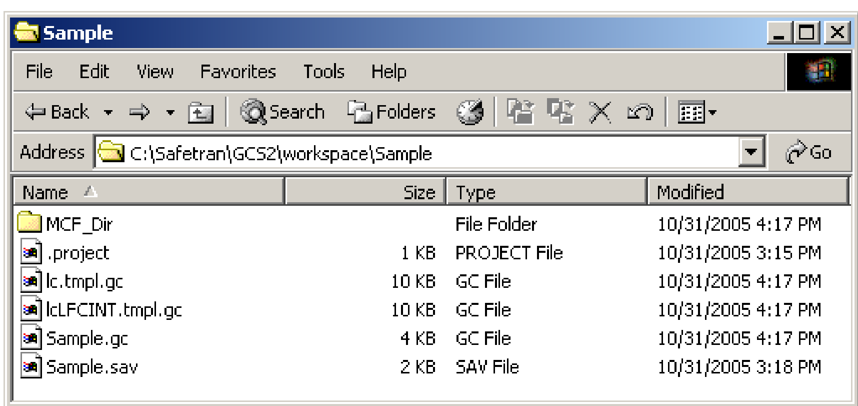
1. Save changes to the Sample.gc file by clicking **File** and selecting **Save**.
2. If errors exist in the Sample.gc file, there will be a red **X** next to the Sample.gc file in the Project View.
3. Click the Problems tab in the Status View to see what the error is in the Sample.gc file.
4. Double-clicking the error in the Status View will take you to the line where the error exists in the Sample.gc file.
5. Correct the error and go back to step 1 until all errors are corrected.

4.19 CREATE A MODULE CONFIGURATION FILE (MCF)

1. In the GCS Projects view, expand **Sample.gc** file tree down to the last level.
2. Select **gweLFCINT**



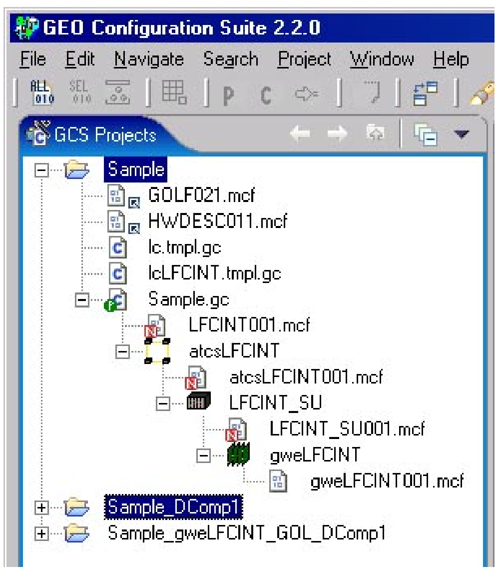
3. Click **Project**
4. Select **Build Selected Projects**
5. The MCF will be placed in a folder called **MCF_Dir** inside the project directory.



4.20 DECOMPILE MCF TO OBTAIN CRC AND COMPARE GC/GL FILES

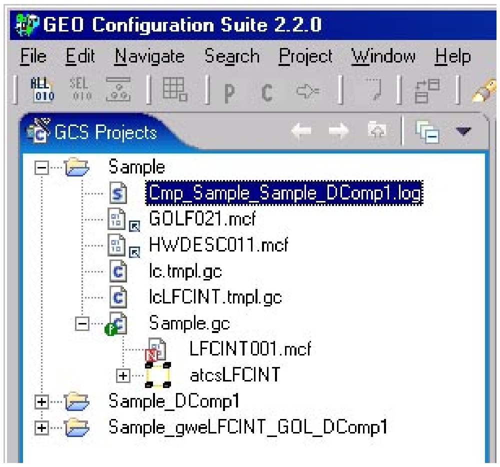
GCS projects should be decompiled, and the decompiled GC/GL files compared to the originals to ensure that all configuration data and vital logic is correct in the MCFs. The source GOLF files must be available to complete the comparison of the GOLF files.

1. In the GCS Projects view, expand **Sample.gc** file tree down to the last level.
2. Select **gweLFCINT001.mcf**
3. Click **Project**
4. Select **Decompile MCF(s)**
5. The MCF, MCF CRC and source GC file will be placed in a folder called **Sample_DComp1** in the **workspace** folder.
6. The GOL source files will be placed in a folder called **Sample_gweLFCINT_GOL_DComp1** in the **workspace** folder.
7. Select both the original project (**Sample)** and the decompiled project (**Sample_DComp1**).



8. Click **Project**
9. Select **Compare Projects**

10. A comparison report called **Cmp_Sample_Sample_DComp1.log** is generated and placed in the original project folder.



<u>The *source* GOLF files must be available to complete the following steps.</u>

1. Create a project for the source GOLF files called **GOLFxxx,** where **xxx** is the version number of the GOLF files.
2. Highlight the new **GOLFxxx** project and import the source GOLF files by using the import tool (Click **File** and select **Import**).
3. Select both the original GOLF project (**GOLFxxx**) and the decompiled GOLF project (**Sample_gweLFCINT_GOL_DComp1**).
4. Click **Project**
5. Select **Compare Projects**
6. A comparison report called **Cmp_GOLFxxx_Sample_gweLFCINT_GOL_DComp1.log** is generated and placed in the original project folder.

GEO SIMULATION

4.21 START SIMULATOR

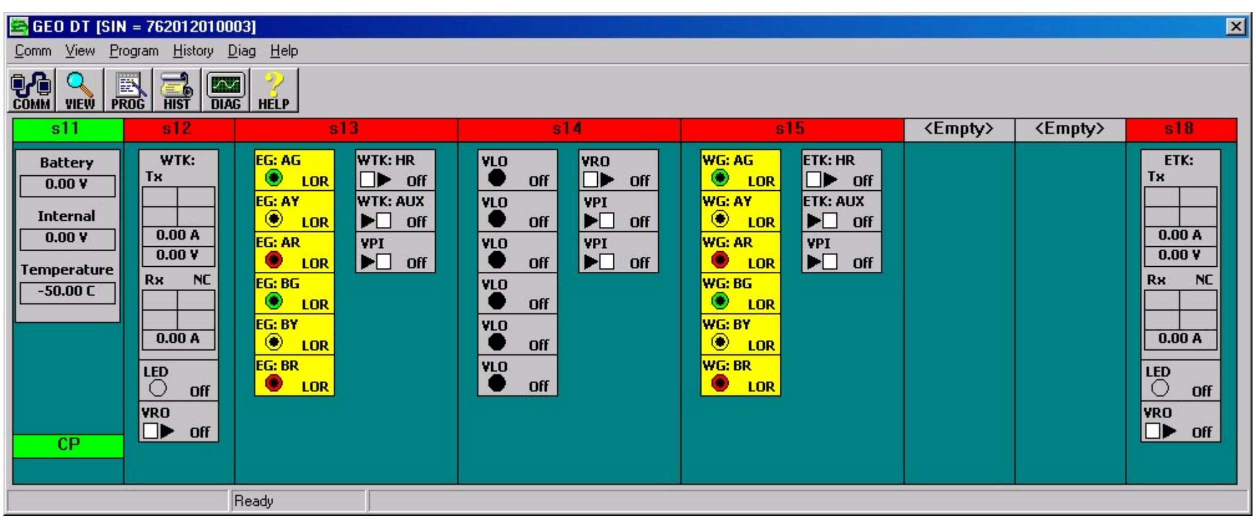
1. In the GCS Projects view, expand **Sample.gc** file tree down to the last level.
2. Select **atcsLFCINT**
3. Click **Project**
4. Select **Run Interactive**

4.22 VIEW AN OBJECT'S SYSTEM STATES WITH GEO DT

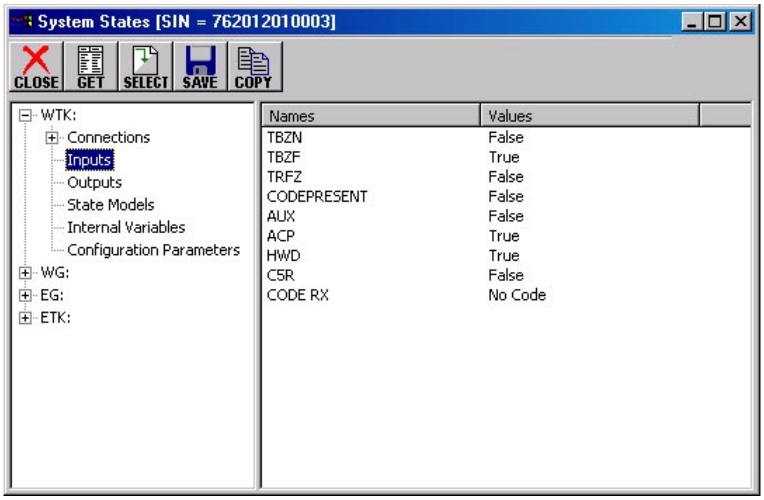
When a simulation is started, it can be monitored using the **GEO DT.**

1. In the **Interactive View**, click the pushbutton that shows the **ATCS address** to start the **GEO DT**.

2. Click **View** on the GEO DT menu bar.
3. Select **System States**



1. In the **System States** window, expand the file tree for the <u>object name</u> you wish to view. (For example, **WTK**).
2. Click to select the appropriate System State whose properties you wish to view. (For example, **Inputs**).
3. Click the **Get** icon on the toolbar to retrieve the current states for that selection.



4.23 SET I/O TO DEFAULT VALUES

In Track Layout window:

1. Right-click on object **WTK** and select **Inputs** > **AUX** > **TRUE**
2. Right-click on object **ETK** and select **Inputs** > **AUX** > **TRUE**
3. Right-click on object **WG** and select **Inputs** > **sig** > **Default**
4. Right-click on object **EG** and select **Inputs** > **sig** > **Default**

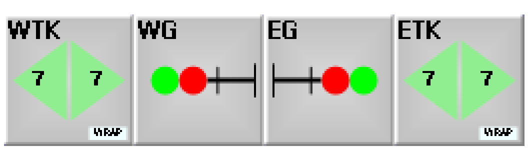
4.24 INPUT VITAL CODE ON CODED TRACKS

NOTE

NOTE

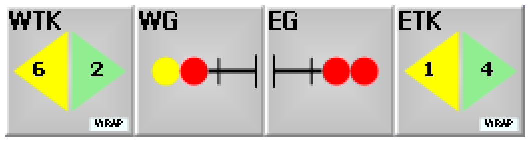
Click anywhere on the Track Layout portion of the screen (Sample.gc) to bring the track layout window to the front to perform the following steps.

1. Right-click on object **WTK** and select **Inputs** > **CODE** > **C7** from the popup menu.
2. Right-click on object **ETK** and select **Inputs** > **CODE** > **C7** from the popup menu.
3. Notice that both WG and EG are displaying aspects of G/R, and that Code 7 is output on both WTK and ETK.



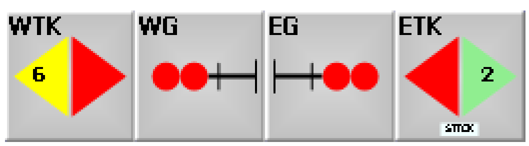
4.25 SETUP FOR WESTBOUND TRAIN MOVEMENT

1. Right-click on object **WTK** and select **Inputs** > **CODE** > **C2** from the popup menu.
2. Notice the aspect of object WG changes to Y/R.
3. Right-click on object **ETK** and select **Inputs** > **CODE** > **C6** from the popup menu.
4. Notice the aspect of object EG changes to R/R.
5. Right-click on object **ETK** and select **Inputs** > **CODE** > **C1** from the popup menu.
6. Verify with the System States window of the GEO DT that WTK shows Inputs – CODE RX: C1, C2 and Outputs – CODE TX: C1, C6. ETK shows Inputs – CODE RX: C1 and Output – CODE TX: C1, C4



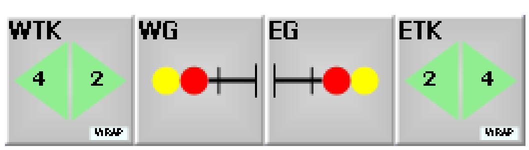
4.26 START WESTBOUND TRAIN MOVEMENT

1. Right-click on object **ETK** and select **Inputs** > **CODE** > **NC** (No Code) from the popup menu.
2. Right-click on object **WTK** and select **Inputs** > **CODE** > **NC** from the popup menu.
3. Notice the aspect of object WG changes to R/R



4.27 COMPLETE WESTBOUND TRAIN MOVEMENT

1. Right-click on object **ETK** and select **Inputs** > **CODE** > **C2** from the popup menu.
2. Right-click on object **WTK** and select **Inputs** > **CODE** > **C1** from the popup menu.
3. Right-click on object **WTK** and select **Inputs** > **CODE** > **C2** from the popup menu.
4. Notice that both WTK and ETK are displaying output of Code 4, and that WG and EG have changed to Y/R.



4.28 END THE SIMULATION

End the simulation by clicking the **X** on **Sample.gc** tab.