In-Vehicle Navigation System Model Deployment Initiative Design Document

Version 1.0

January 7, 1998

SwRI Project No. 10-8684
P.O. No. 7-70030
Req. No. 60115-7-70030

Prepared For:

Texas Department of Transportation
TransGuide
3500 NW Loop 410
San Antonio, Texas 78229

Prepared by:

Southwest Research Institute
P.O. Drawer 28510
San Antonio, Texas 78228

Approval Page

In-Vehicle Navigation Project Manager	Date
SwRI MDI Project Manager	Date
Communications Engineering Dept. Director	Date
TxDOT Approval	Date

Acronym List

ATMS	Advanced Traffic Management System
AVI	Automatic Vehicle Identification
DGPS	Differential GPS
DS	MDI Data Server
GPS	Global Positioning System
ITS	Intelligent Transportation System
IVN	In-Vehicle Navigation
LR	Location Reference
MCS	Master Computer Software
MDI	Model Deployment Initiative
PCB	Printed Circuit Board
STIC	Subcarrier Traffic Information Channel
STM	STIC Transmission Message
SwRI	Southwest Research Institute
TIM	Traffic Information Message
TMC	Traffic Management Center
TOC	TransGuide Operations Center
TxDOT	Texas Department of Transportation

Table of Contents

List of Figures

1. Introduction

Real-time in-vehicle navigation is a significant traveler information service added to the TransGuide system during the Model Deployment Initiative (MDI). The TransGuide In-Vehicle Navigation (IVN) system includes an IVN unit that provides travelers with route guidance, vehicle position, and regional points of interest information. Units currently available on the market utilize a database of transit times for each road segment to calculate the shortest travel time from where the vehicle is located to a point the driver inputs as the destination. These databases are static, as they contain typical transit times that by definition cannot account for real-time congestion caused by traffic accidents, rush hours, public events, etc. The MDI implementation of IVN addresses this shortcoming by providing the units with real-time traffic data. The system includes the infrastructure necessary to deliver the real-time information to vehicles in the San Antonio area and the initial deployment of 590 IVN units that utilize the real-time traffic data.

1.1 Purpose of System

The purpose of the IVN system is to communicate real-time traffic information maintained at the TransGuide Traffic Management Center (TMC) to travelers in the San Antonio metropolitan area and present the information in a manner useful for making navigation decisions.

1.2 Operational Concept

Several types of traffic speed information are collected by the TransGuide system: actual highway traffic speed collected by loop detectors; actual highway and arterial traffic speed collected by the automatic vehicle identification (AVI) detection system; predicted arterial traffic speed based on observation of traffic over a period of time; traffic incident information collected by the San Antonio Police Department; daily scheduled road/lane closures; and railway crossing information. This information is stored and maintained by the MDI Data Server (DS) at the TransGuide TMC.

The IVN System consists of the following four major components:

- IVN master computer,
- Subcarrier Traffic Information Channel (STIC) message encoder,
- STIC receiver,
- In-vehicle navigation unit.

The IVN master computer obtains information from the Data Server and formats the information into efficient messages. The master computer then communicates the data over a dial-up modem connection to the STIC message encoder.

The STIC message encoder and STIC receiver comprise a commercially available data broadcast system. The STIC message encoder is located in the transmission room of a commercial FM radio station. The encoder receives messages from the master computer and encodes them into a baseband signal that is modulated and broadcast as a sub-carrier to the FM radio signal. The FM subcarrier may coexist with other subcarriers on the same FM radio signal.

The STIC receiver and a commercially available in-vehicle navigation unit are installed in vehicles in the San Antonio metropolitan area. The STIC receiver receives the FM radio signal

and decodes the subcarrier. The STIC receiver passes the messages to the navigation unit over a serial data link.

The navigation unit decodes the messages, displays the real-time information along with the map data to the traveler(s) in the vehicle, and calculates the quickest route to a traveler-entered destination using the real-time and map data.

The operational concept is illustrated in Figure 1.

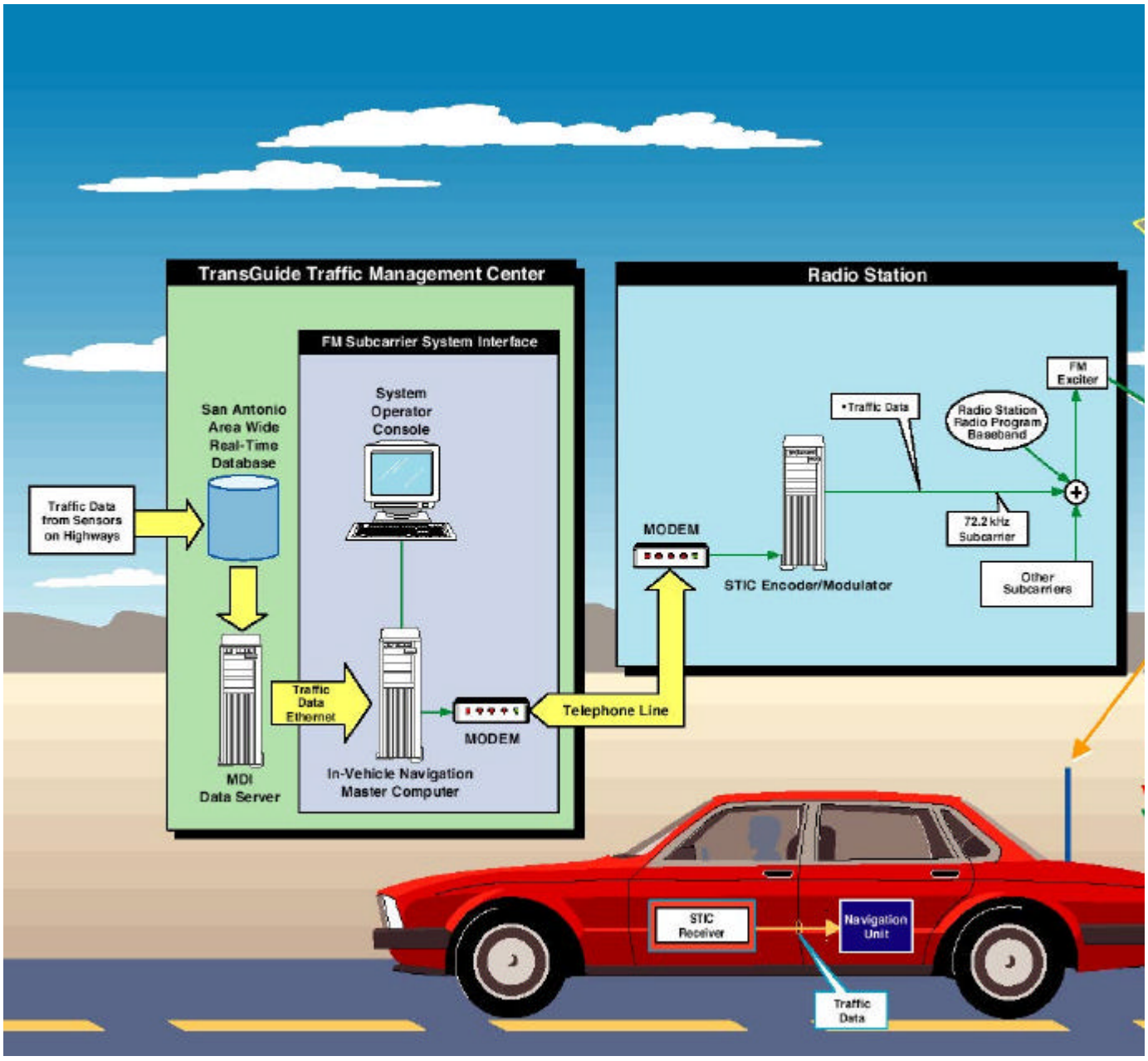

Figure 1. Operational Concept

1.3 System Objectives

Three objectives of the IVN system are:

- to enable San Antonio area travelers to reach destinations more quickly by avoiding traffic congestion,
- to reduce San Antonio area traffic congestion by guiding travelers through uncongested roadways, and
- to increase and demonstrate the effectiveness of the TransGuide ITS.

1.4 Referenced Documents

The following documents are referenced by this design document.

- Texas Department of Transportation, *Request for Offer (RFO) for the Model Deployment Initiative System Integration, 60115-7-70030*, Specification No. TxDOT 795-SAT-01, October, 1996.
- Southwest Research Institute, *Proposal for the 6015-7-70030: Request for Offer, Model Deployment Initiative System Integration*, SwRI Proposal No. 10-20342, November, 1996.
- Southwest Research Institute, *In-Vehicle Navigation System Model Deployment Initiative Preliminary Design Document, Version 1.0*, February 14, 1997.
- Southwest Research Institute, *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol, Version 1.0*, May 29, 1997.
- Southwest Research Institute, *In-Vehicle Navigation System Model Deployment Initiative Version Description Document, Version 1.0*.
- Southwest Research Institute, *Automated Vehicle Identification System Model Deployment Initiative Design Document, Version 1.0*.
- Scientific-Atlanta, *Subcarrier Traffic Information Channel Interface Control Document, Revision A*.

2. External Interfaces

The IVN system externally interfaces with the MDI Data Server, public switched telephone network, an FM radio station, and many vehicles. These interfaces are illustrated in Figure 2. A detailed description of each interface is included in Section 4.

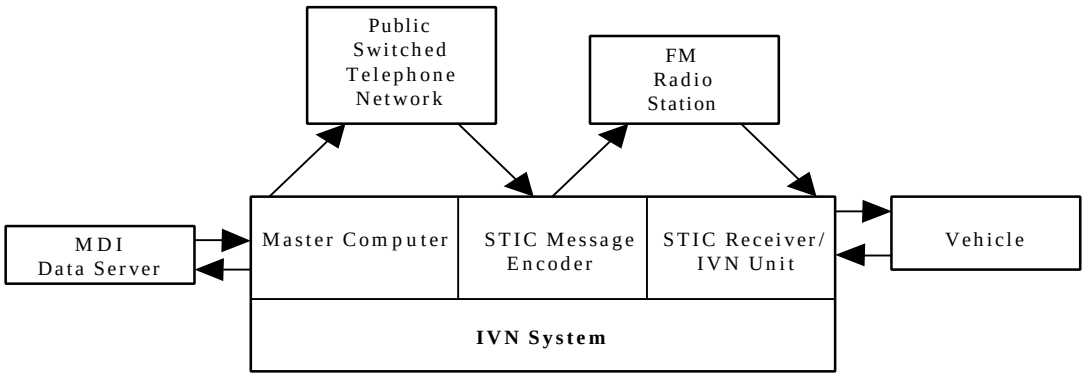

Figure 2. IVN System External Interfaces

3. System Requirements

The following IVN system requirements are derived from the *Request for Offer for the Model Deployment Initiative System Integration* written by TxDOT and the *Proposal for the Request for Offer for the Model Deployment System Integration* written by SwRI.

3.1 System Level Requirements

IVN-1 The system shall communicate the following types of real-time traffic information to moving vehicles:
- link speed data,
- incident information.

IVN-2 The real-time traffic information shall be derived from the Data Server.

IVN-3 The system shall communicate the real-time traffic information using the Subcarrier Traffic Information Channel (STIC) system that consists of a message encoder/FM subcarrier generator and FM subcarrier receivers.

IVN-4 The system shall utilize commercially available IVN units to display the real-time traffic information to travelers.

IVN-5 The system shall utilize the IVN units of at least two manufacturers.

3.2 Master Computer Requirements

The following requirements are subordinate to requirement IVN-2.

IVN-2.1 Software running on an IVN master computer shall be used to extract real time information from the Data Server.

IVN-2.2 The IVN master computer shall be a Sun Microsystems Ultra SPARCStation with the following components:
- 167 MHz SPARC (RISC) CPU,
- 128 MB RAM,
- 4.2 GB hard disk space,
- Floppy disk drive,
- Sun CD-ROM drive,
- Turbo GX+ graphics,
- 20" color monitor,
- 8 port modem server (SCSI attached),
- Dual ethernet interfaces, and
- Dual SCSI channels.

IVN-2.3 The master computer shall be located in the TransGuide Operations Center.

3.3 IVN Unit Requirements

The following requirements are subordinate to requirement IVN-4.

IVN-4.1 The IVN unit shall be composed of the following components:
- microprocessor,
- reconfigurable LCD color display,
- removable media for data storage,

- gyroscopic sensor, and
- GPS receiver.

IVN-4.2 The IVN unit shall accept real-time traffic data input from the STIC receiver.

IVN-4.3 The IVN unit shall provide a means for the traveler to enter a destination of the following types:
- address,
- intersection,
- place or point of interest, and
- previous destination.

IVN-4.4 The IVN unit shall calculate the shortest time route, based on the real-time information from the STIC receiver, from the current location of the vehicle to the traveler-entered destination.

IVN-4.5 The IVN unit shall communicate the route information using a map display, guide display, and audible prompting.

IVN-4.6 The IVN unit guide display shall present the following information:
- distance to an upcoming turn, and
- direction of an upcoming turn.

IVN-4.7 The IVN unit shall provide the following audible prompts:
- warning of an upcoming turn, and
- indication to make a turn.

IVN-4.8 The IVN unit map display shall show the current location of the vehicle and the calculated route on an annotated map of the surroundings.

IVN-4.9 The information used to generate the map display shall be derived from the San Antonio, Texas Metropolitan Area database supplied by Navigation Technologies.

3.4 Master Computer Software Requirements

The IVN master computer software (MCS) requirements in this section are subordinate to requirement IVN-2.1. These requirements are derived from software design decisions described in Section 4.2 and were used to guide the development of software subcomponents.

IVN-2.1.1 The IVN MCS shall transmit IVN process status information to the Data Server every 60 seconds.

IVN-2.1.2 The IVN MCS shall extract link ID information from the Data Server Realtime Subsystem.

IVN-2.1.3 The IVN MCS shall extract link location information from the Data Server Realtime Subsystem.

IVN-2.1.4 The IVN MCS shall extract link speed information from the Data Server Realtime Subsystem.

IVN-2.1.5 The IVN MCS shall extract incident information from the Data Server Realtime Subsystem.

IVN-2.1.6 The IVN MCS shall transmit link speed information to the STIC message encoder every 60 seconds.

IVN-2.1.7 The IVN MCS shall transmit incident information to the STIC message encoder every 60 seconds.

IVN-2.1.8 The IVN MCS shall transmit locally referenced link speed information to the STIC message encoder in accordance with the messaging protocol defined in the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol*.

IVN-2.1.9 The IVN MCS shall transmit globally referenced link speed information to the STIC message encoder in accordance with the messaging protocol defined in the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol*.

IVN-2.1.10 The IVN MCS shall transmit locally referenced incident information to the STIC message encoder in accordance with the messaging protocol defined in the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol*.

IVN-2.1.11 The IVN MCS shall transmit globally referenced incident information to the STIC message encoder in accordance with the messaging protocol defined in the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol*.

IVN-2.1.12 The IVN MCS shall generate a table of TransGuide link information that includes link ID, starting coordinate, ending coordinate, and street name.

IVN-2.1.13 The IVN MCS shall provide a display of the process status, which shall be updated every 60 seconds.

IVN-2.1.14 The IVN MCS shall be capable of being started and stopped through the IVN process status GUI.

IVN-2.1.15 The IVN MCS shall provide a display of the status of communication to peripheral systems with which it exchanges data.

IVN-2.1.16 The IVN MCS shall log informational messages, warning messages, and error messages to a status log file.

4. System Design

The system design is described in the following two sections. The system architecture section includes a description of the hardware components and their installation and interconnection. The software design section describes the design of the master computer software.

4.1 System Architecture

The system architecture is documented in a set of engineering drawings that accompany this design document. Reduced size copies of these drawings are included in Appendix A as a reference for the reader. This section of the design document provides a description of the system architecture that is illustrated in the drawing package.

The system consists of four sub-systems: the IVN master computer, the STIC message encoder, the STIC receiver, and the IVN unit. Drawing number 8684-5000 is a system level block diagram that indicates the associated drawing and location for each of the four sub-systems.

4.1.1 IVN Master Computer

The IVN master computer executes software that extracts real-time traffic information from the Data Server. The software encapsulates the real-time traffic data into messages that are delivered by the STIC transmission system to the IVN unit. The format and content of the messages adhere to the communications protocol described in the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol* document. The software sends the real-time traffic information messages to the STIC message encoder using a modem server and standard dial-up telephone line. The IVN master computer software design is described in Section 4.2.

The IVN master computer is located in the "computer room" of the TxDOT TransGuide Operations Center (TOC). The master computer and associated components are mounted in a 23" equipment rack. Drawing number 8684-5003 shows the equipment part numbers, rack configuration, interconnection of components, and external connections. This drawing also shows the components of the Automated Vehicle Identification (AVI) system master computer. Refer to the *Automated Vehicle Identification Model Deployment Initiative Design Document* for further information about that computer.

The IVN master computer is a Sun Microsystems Ultra SPARCStation with the following components:

- 167 MHz SPARC (RISC) CPU,
- 128 MB RAM,
- Two 2.1 GB hard disks,
- Floppy disk drive,
- CD-ROM drive,
- Turbo GX+ graphics interface,
- 20" color monitor,
- Dual ethernet interfaces,
- Dual SCSI channels, and
- 16 port modem server (SCSI attached).

The IVN master computer external electrical interfaces include an ethernet connection, modem connection, and power connections. The IVN master computer communicates with the Data Server over an ethernet network shared by other workstations at TransGuide. The IVN master computer communicates with the STIC message encoder over a dial-up modem connection that is achieved by connecting a standard switched telephone line to the modem server. The IVN master computer, monitor, and modem server each obtain electrical power from a standard 110VAC 60Hz power outlet.

Additional information on the detailed operation and maintenance of the Sun SPARCStation, monitor, and modem server may be obtained from the operator manuals which accompany those pieces of commercial equipment.

4.1.2 STIC Message Encoder

The STIC message encoder, manufactured by Scientific-Atlanta, receives the messages sent by the IVN master computer over the modem connection, and depending on the command from the master computer, the encoder adds or deletes messages from a message queue. The encoder modulates the messages in the queue into a baseband signal that is centered at 72 Khz and outputs this baseband signal to the FM exciter of the radio station. The FM exciter adds the baseband signal from the STIC encoder to the regular audio program and other subcarriers. The composite signal is then modulated by the FM transmitter and broadcast over the entire San Antonio metropolitan area.

The STIC message encoder optimizes the data transmission throughput by embedding forward error correction information in the messages. A 15-byte Reed-Solbman forward error correction code accompanies every 228 bytes of broadcast data. In addition, the encoder continuously cycles through the message queue, rebroadcasting messages that are not deleted by the master computer. This rebroadcast also improves transmission success.

The encoder is located at the Tower of the Americas in the radio transmission room of KTFM 102.7FM. The encoder and associated components are mounted in a 19" equipment rack. Drawing number 8684-5004 shows the equipment part numbers, rack configuration, interconnection of components, and external connections.

The STIC message encoder is a rack mountable environmentally hardened IBM compatible PC with the folbwing components:

- 486 66MHz CPU,
- 28.8kbps internal modem,
- STIC subcarrier modulator ISA board,
- GPS receiver,
- GPS antenna,
- hard disk drive,
- 3.5" floppy drive,
- Built-in monitor,
- rack mount keyboard, and
- rack mount surge protector.

The STIC message encoder external electrical interfaces include a subcarrier output, modem connection, and power connection. The subcarrier output is connected to an SCA input of the FM exciter of the radio station by a coaxial cable. The encoder communicates with the IVN master computer by connecting the internal modem to the public switched telephone network.

The STIC message encoder obtains electrical power from a standard 110 VAC 60Hz power outlet provided by the accompanying surge protector.

The software that is executed by the STIC message encoder was also provided by Scientific-Atlanta; however, SwRI enhanced this software by adding a batch file that configures the message encoder for proper and reliable operation. When the message encoder is booted, the batch file automatically configures the modem and starts the STIC message encoder software. The *In-Vehicle Navigation System Model Deployment Initiative Version Description Document* describes the proper configuration of the message encoder including the message encoder software and SwRI added batch file.

The message encoder software has only one tunable parameter. This parameter controls the magnitude of the 72Khz subcarrier signal. This parameter can be used to adjust the injection level or "deviation" of the subcarrier. This parameter is modified by typing an uppercase "G" at the message encoder keyboard and using the up and down arrow keys for the adjustment. A deviation of 10% is recommended by Scientific-Atlanta for optimum performance. This is the signal level that the message encoder is currently injecting into the KTFM broadcast signal.

The message encoder also includes a GPS antenna and receiver that is used by the encoder to generate differential GPS data. This DGPS data is broadcast in the subcarrier along with the traffic data, but as a different data stream. The DGPS data is separated from the traffic data by the STIC receiver in the vehicle and is not processed by the IVN units. This feature of the STIC message encoder is not used by the IVN system.

Additional information on the detailed operation and maintenance of the STIC message encoder may be obtained from the accompanying documentation supplied by Scientific-Atlanta.

4.1.3 STIC Receiver

The STIC receiver, also manufactured by Scientific-Atlanta, continuously searches all FM radio frequencies for a station broadcasting the STIC subcarrier signal. When the receiver tunes to 102.7FM, it finds the STIC subcarrier and begins to receive the messages broadcast by the message encoder. The receiver uses the forward error correction codes transmitted by the encoder to correct transmission errors. After error correction, the receiver sends the messages to the IVN unit over an RS-232 serial data link.

The time required by the STIC receiver to search through the FM frequencies can vary. The receiver is specially designed to find 102.7FM quickly, and in the presence of a strong FM signal, the receiver will begin to decode data within 30 seconds. If the receiver loses lock due to a weak FM signal, it will regain lock within 30 seconds from the time that the FM signal is re-established. In the event that the broadcast signal moves to another frequency, the search will be much slower. The STIC receiver could require as long as 3 minutes to find a subcarrier on a frequency other than 102.7FM.

The STIC receiver's ability to lock onto the broadcast signal is dependent upon the quality of the signal at the receiver input. Several factors can affect the signal quality. The strength of the receiver input signal is greatly dependent upon the type of antenna used, the mounting of that antenna, and antenna position. For good reception of the STIC subcarrier, it is important that a quality antenna is used and is properly installed, including a connection to a sufficient ground plane.

For convenience and aesthetics, it is often desirable to input the same signal from the existing vehicle FM antenna to both the STIC receiver and existing FM radio using a power splitter.

This approach, however, results in reduced signal strength to the STIC receiver and can affect the receiver's ability to lock onto the broadcast signal.

The location of the receiver is also an important factor for reception. Although the KTFM signal approaches an omnidirectional coverage pattern, the signal does not cover the San Antonio area uniformly. The primary lobe of the KTFM signal is centered in a southwesterly direction from the Tower of Americas, and reception of the STIC subcarrier will be best on that side of the city. In addition, the specific electromagnetic environment of any particular site may affect data reception. This is especially true of sites that are within large structures containing metal (such as commercial buildings and parking garages) or in the "shadow" of such structures. At sites where reception is difficult, moving the receiver antenna by only a few feet can make the difference between good and bad data reception.

In any case, the reception of the STIC subcarrier should not be compared with the reception of the KTFM audio program using a FM radio. The carrier signal that contains the audio program is ten times stronger than the STIC subcarrier. The audio program is also closer to the FM pilot signal, which greatly improves reception. For these reasons, the STIC receiver will not receive data in some places where the audio program is clearly received.

The STIC receiver is typically mounted in a vehicle adjacent to the controller of the IVN unit. Drawings 8684-5001 and 8684-5002 are installation drawings that show typical installations of the receiver and IVN units in a passenger vehicle.

The electrical interfaces of the STIC receiver include an FM antenna, data, power, and ground connection. The receiver either shares the signal from an existing FM antenna with the radio using a power divider, or is connected directly to a dedicated FM antenna. For data communication, an RS-232 cable connects to a 9 pin RS-232 connector on the receiver and similar connector on the IVN unit interface. Also required are switched power and ground connections that are typically made with vehicle ignition circuit.

The STIC receiver also decodes the DGPS data broadcast by the message encoder. The DGPS data is output on two unused pins of the RS-232 serial link; however, this data is not processed by the IVN units.

Additional information on the detailed operation and maintenance of the STIC receiver may be obtained from the accompanying documentation supplied by Scientific-Atlanta.

4.1.4 IVN Unit

The IVN unit is one of two commercially available products. Alpine Electronics Research of America and Zexel USA each supply a navigation unit capable of receiving and using real-time traffic data from the IVN system.

Both the Alpine and Zexel navigation units include a controller and display head. The controller houses a microprocessor, GPS receiver, removable PCMCIA hard disk or CD-ROM, and gyroscopic sensor. The display head on the Zexel IVN unit has keys that the traveler may use to enter information. The display head on the Alpine unit has an infrared sensor and is accompanied by a handheld remote control that provides the traveler a means for data input.

The IVN units accept the real-time traffic messages from the STIC receiver. The messages are unwrapped by the navigation unit software, and the real-time information is incorporated into a map database maintained by the units. The navigation units display the map database information and real-time data to travelers on the LCD display.

Both the Zexel and Alpine IVN units have graphical user interfaces that allow destination input by address, intersection, point of interest, or previous destination. The Alpine unit also allows the traveler to select a destination by map input.

The IVN units calculate a route from the current location of the vehicle to the traveler-entered destination. The traveler may select a computation of the shortest time route or the route that includes the least use of freeways. The IVN units calculate the shortest time route using the real-time information broadcast by the IVN system. Additionally, the Alpine IVN unit provides route calculations for the fewest turns and least tollways.

The IVN units present the real-time traffic and route information using a map display, guide display, and audible messaging. The map display shows a map of the area around the present location of the vehicle. The map display includes geometry of roadways, street names, and an icon indicating the current location of the vehicle. The traveler may zoom in or out to achieve the level of detail in the display that he requires. The map display highlights the roads appearing at the current zoom level that are on the calculated route and highlights areas of traffic congestion by color coding road segments.

The guide display provides turn-by-turn navigation directions to the traveler. The guide display indicates the direction of an upcoming turn with a large arrow pointing in the direction of the turn. The guide display also presents the distance to the upcoming turn as determined by the calculated route and vehicle position.

The IVN units generate audible messages in the form of a voice prompt indicating the distance and direction of an upcoming turn. When a turn is pending, another audible prompt signals the driver to make the turn.

The IVN units determine the location and heading of the vehicle using a GPS receiver, a gyroscopic sensor, and map matching. The GPS receiver provides coarse position information that is compared to map data, then the measured position is "snapped" to the map using map matching. The gyroscopic sensor provides accurate vehicle heading information to the IVN units.

The electrical interfaces of the IVN units include GPS antenna, 9 pin RS-232, speed pulse, back-up light, ignition, power, and ground connections. The connecting cable of the GPS antenna, which accompanies the IVN unit, plugs into a jack on the IVN unit CPU. For traffic message communication, an RS-232 cable connects to a 9 pin RS-232 connector on an interface to the IVN unit and similar connector on the STIC receiver. The IVN unit has a wire harness that taps into the vehicle speed pulse near the powertrain control module. In addition, the IVN unit wire harness taps into the back-up light, ignition, power, and ground in the most convenient place near the unit. The location of these connections vary greatly from vehicle to vehicle. Drawings 8684-5001 and 8684-5002 are installation drawings that show typical installations of the IVN units and STIC receiver in a passenger vehicle.

Additional information on the detailed operation and maintenance of the IVN units may be obtained from the accompanying documentation supplied by Alpine and Zexel.

4.2 IVN Master Computer Software Design

The function of the IVN Master Computer Software (MCS) is to extract real-time data from the MDI Data Server and transmit the data to the STIC message encoder, which broadcasts the data via FM radio to the receivers and IVN units in vehicles. The IVN MCS runs on a Sun Microsystems Ultra SPARCStation. The hardware components of the computer system are

described in Section 4.1.1. The configuration of the IVN Master Computer and MCS are described in the *In-Vehicle Navigation System Model Deployment Initiative Version Description Document.* The architecture of the IVN MCS is described in the following sections and includes descriptions of the external interfaces, the high level system design, the low level functional decomposition, and the format of the primary data structures.

4.2.1 External Interfaces

The IVN MCS runs as a single UNIX process. This process has five external interfaces as depicted in the figure below.

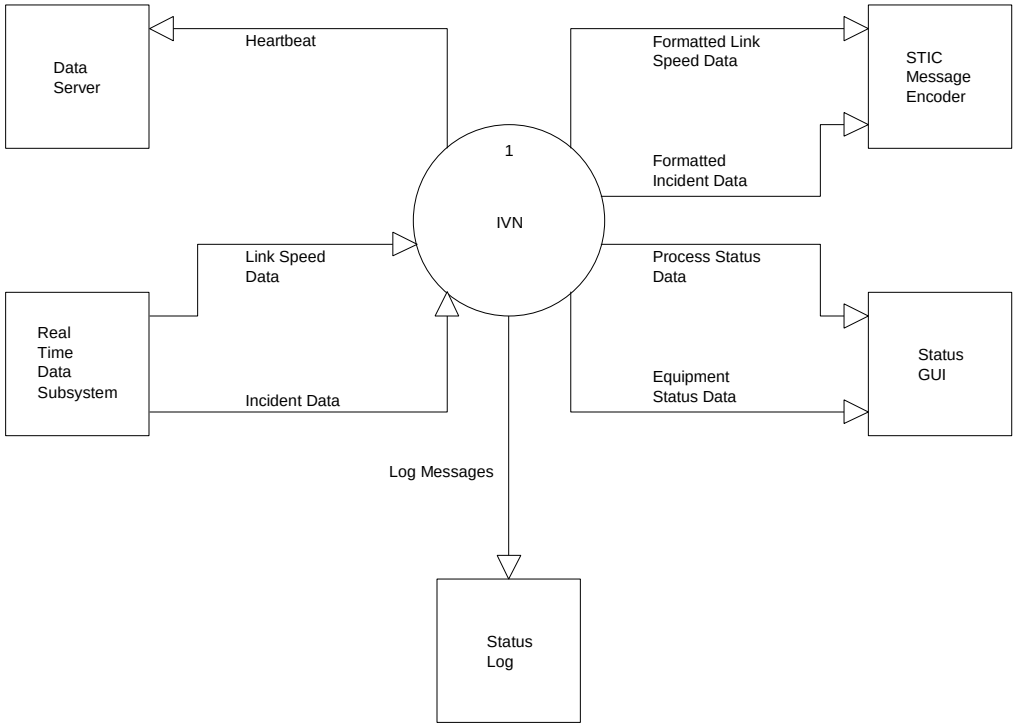

Figure 3. IVN MCS External Interfaces

The external entities to which the IVN process communicates include the Data Server which is a process running on another workstation, the Real Time Data Subsystem which is a process running on the IVN Master Computer, the Status GUI which is a process running on the IVN Master Computer, the Status Log which is a text file maintained on the IVN Master Computer fixed disk, and the STIC message encoder which is a PC installed at an FM radio station. These are described further in the following table.

External Entity	Description
Data Server	The Data Server is the central repository of information generated and maintained by the MDI subsystems. The IVN process sends its process status information to the Data Server at regular intervals.
Real Time Data Subsystem	The Real Time Data Subsystem provides the current speed and location for each of the TransGuide links. It also provides the type, start time, and location of each currently active incident.
Status GUI	The status GUI process is the graphical user interface which provides the visual description of the IVN process. The status GUI indicates the overall process status and also provides a detailed status of the peripheral equipment with which the IVN process communicates. The IVN process can also be started or stopped through the status GUI.
Status Log	The Status Log is a text file stored on the IVN Master Computer fixed disk drive which contains timestamped log messages for the IVN process. A log file for each day of the week is maintained and files are kept for the current week. The status log can be viewed with a text editor.
STIC Message Encoder	The STIC message encoder is a PC installed in the transmission room of a commercial FM radio station. The STIC message encoder incorporates hardware and software designed to receive messages from the IVN Master Computer and encode them into a baseband signal that is modulated and broadcast as a sub-carrier to the FM radio signal. The STIC message encoder communicates to the IVN Master Computer via dial-up modem.

The data flows between the IVN process and the external entities to which it communicates are described in the following table.

Data Flow	Description
Equipment Status Data	The IVN equipment status data describes the status of communication to external devices including the Real Time Data Subsystem, the modem, and the STIC message encoder. The status of each device is defined to be either OKAY, ERROR, or UNKNOWN. The IVN equipment status information is transmitted to the Status GUI process through shared memory.
Formatted Incident Data	The formatted incident data includes the incident type code (major collision, minor collision, etc.), the incident location (latitude, longitude and level or link ID), and the incident start time of each active incident formatted according to the IVN communication protocol. The IVN process transmits the incident data to the STIC Message Encoder through dial-up modem interface as defined by the STIC communication protocol.
Formatted Link Speed Data	The formatted link speed data includes the current speed (in meters per second) and the location (starting and ending latitude, longitude and level) of each of the active TransGuide links formatted according to the IVN communication protocol. The IVN process transmits the link speed data to the STIC Message Encoder through dial-up modem interface as defined by the STIC communication protocol.
Hearbeat Data	The Heartbeat Data is the IVN process status which is transmitted to the Data Server at regular intervals. The IVN process status is defined to be either OKAY, WARNING, or ERROR. The Heartbeat Data is transmitted to the Data Server via a socket interface.
Incident Data	The incident data includes the incident source code (ATMS, SAPD, etc.), the type code (major collision, minor collision, etc.), the incident location (latitude, longitude and level or link ID), and the incident start time of each currently active TransGuide incident. The data is communicated to the IVN process through a socket interface.
Link Speed Data	The link speed data includes the current speed (in miles per hour) and the location (starting and ending latitude, longitude and level) of each of the TransGuide links. The data is communicated to the IVN process through a socket interface.
Log Messages	Log messages are text strings which are written optionally to the status log file and/or the standard output. The log messages include the message type (debug, informational, warning, or error), the message timestamp, the module (function) reporting the message, and the message text.
Process Status Data	The IVN process status data includes the current IVN process status which is defined to be either OKAY, WARNING, or ERROR. The IVN process information is transmitted to the Status GUI process through a shared memory interface.

4.2.2 System Design

The IVN software runs as a single process. This process has four distinct components or stages of operation as described below: startup, main loop, shut down, and asynchronous signal handling.

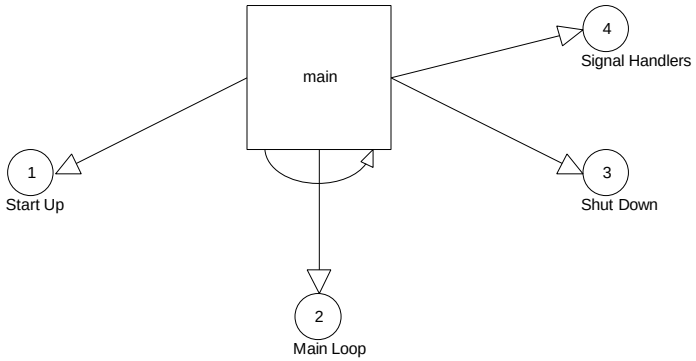

Figure 4. IVN Main Procedure

Component	Description
Startup	Process initialization and startup operations for main loop.
Main Loop	Main loop which continuously reads and transmits real time data to STIC.
Shutdown	Process shutdown operations and program exit.
Signal Handling	Asynchronous signal handlers.

4.2.2.1 Startup

The startup procedure includes the IVN process initialization operations and the allocation and initialization of the data structures for maintaining the real time data and communicating it to the STIC system.

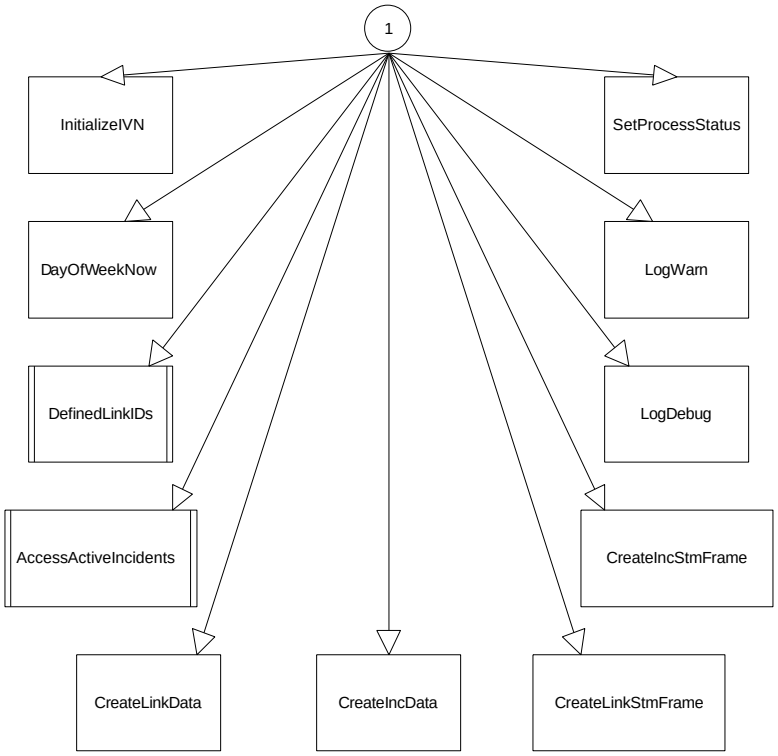

Figure 5. IVN Initialization and Startup

Function	Description
AccessActiveIncidents	Real Time Data library function. Returns a pointer to the list of currently active TransGuide incidents.
CreateIncData	Creates a local copy of the TransGuide incident data.
CreateIncStmFrame	Creates the array of incident STMs which contain the incident type and location for a predefined number of incidents, formatted according to the IVN communication protocol.
CreateLinkData	Creates a local copy of the TransGuide link data.
CreateLinkStmFrame	Creates the array of link speed STMs which contain the link speed and location for each of the TransGuide links, formatted according to the IVN communication protocol.
DayOfWeekNow	Returns the current day of week.
DefinedLinkIDs	Real Time Data library function. Returns a pointer to the list of TransGuide link IDs.
InitializeIVN	Performs the operations required to initialize the IVN process.
LogDebug	Logs debug messages to the status log and/or standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).

4.2.2.2 Main Loop

Upon completion of the startup procedure, the IVN process begins executing the main bop. The main bop is the normal operational mode of the IVN process. The basic operations

performed by the main loop include reading and updating the real time data, formatting the data according to the IVN communication protocol, and transmitting the formatted data to the STIC system. The main loop runs continuously until it is stopped by the user (by termination signal) or a fatal error occurs.

Figure 6. IVN Main Loop

Function	Description
AccessActiveIncidents	Real Time Data library function. Returns a pointer to the list of currently active TransGuide incidents.
CheckModemConnection	Checks the connection between the IVN and STIC modems and establishes the connection if it is down.
DayOfWeekNow	Returns the current day of week.
InitIncData	Initializes the local copy of the TransGuide incident data array with the current incident data and determines which incidents are locally referenced and which are globally referenced.
IvnProcessStopped	Returns true if the IVN process has been stopped.
LogDebug	Logs debug messages to the status log and/or standard output.
LogInfo	Logs informational messages to the status log and/or standard output.
LogRealTimeData	Logs the current values of the real time link speed and incident data to the status log and/or standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
RenumberStmFrame	Re-numbers the STM sequence numbers in an STM array.
ResetSequenceNumber	Resets the STM sequence number counter.
SendStmFrame	Transmits either a link or incident STM data array to the STIC system.
SetFirstStm	Sets a field in the header of an STM to mark it as the first STM in the superframe of STMs.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
SetWatchDog	Sets the IVN process watchdog timer flag.
SticMessage	Issues a message to the STIC system and reads and verifies the STIC response.
UpdateIncStmFrame	Updates the incident STM array with the current incident data.
UpdateLinkStmFrame	Updates the link speed STM array with the current link speed data.
UpdateRealTimeData	Refreshes the real time data and checks the timestamp of the data to verify that live data is being received.

4.2.2.3 Shutdown

The shutdown procedure is executed upon termination of the main loop, either due to user command or fatal error. The shutdown procedure consists of closing communication connections, updating the process status, and exiting the program.

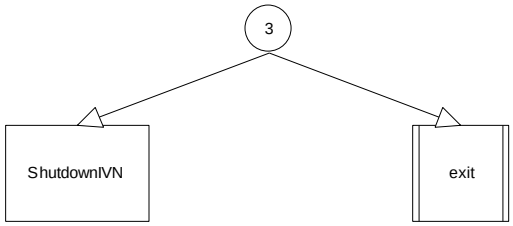

Figure 7. IVN Shutdown

Function	Description
exit	C library function.
ShutdownIVN	Performs the operations required to gracefully shut down the IVN process.

4.2.2.4 Signal Handlers

The IVN signal handlers capture and respond to the asynchronous signals received by the IVN process. Special handlers are defined for the Data Server heartbeat timer signal and process termination signal which are the assigned signals for the IVN process. Unassigned signals are handled by a general purpose signal handler.

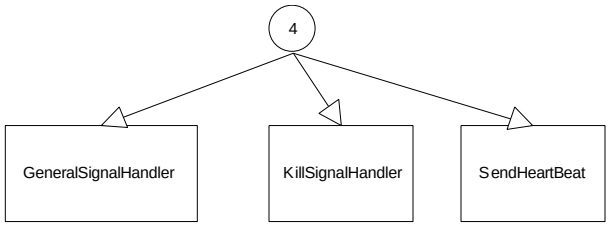

Figure 8. IVN Signal Handlers

Function	Description
GeneralSignalHandler	Catches and handles unassigned signals.
KillSignalHandler	Catches and handles the SIGTERM signal which causes the IVN process to gracefully shutdown.
SendHeartBeat	Timer handler for Data Server heartbeat timer. Transmits the IVN process status to the Data Server.

4.2.3 Subsystem Design

The subsystem design describes the lower level component functions of the IVN software.

4.2.3.1 Initialization

The IVN initialization procedure executes once at startup and performs a number of functions including assigning signal handlers, loading configurable parameters, opening the status log, and initializing communication interfaces. The initialization procedure and its component functions are described below.

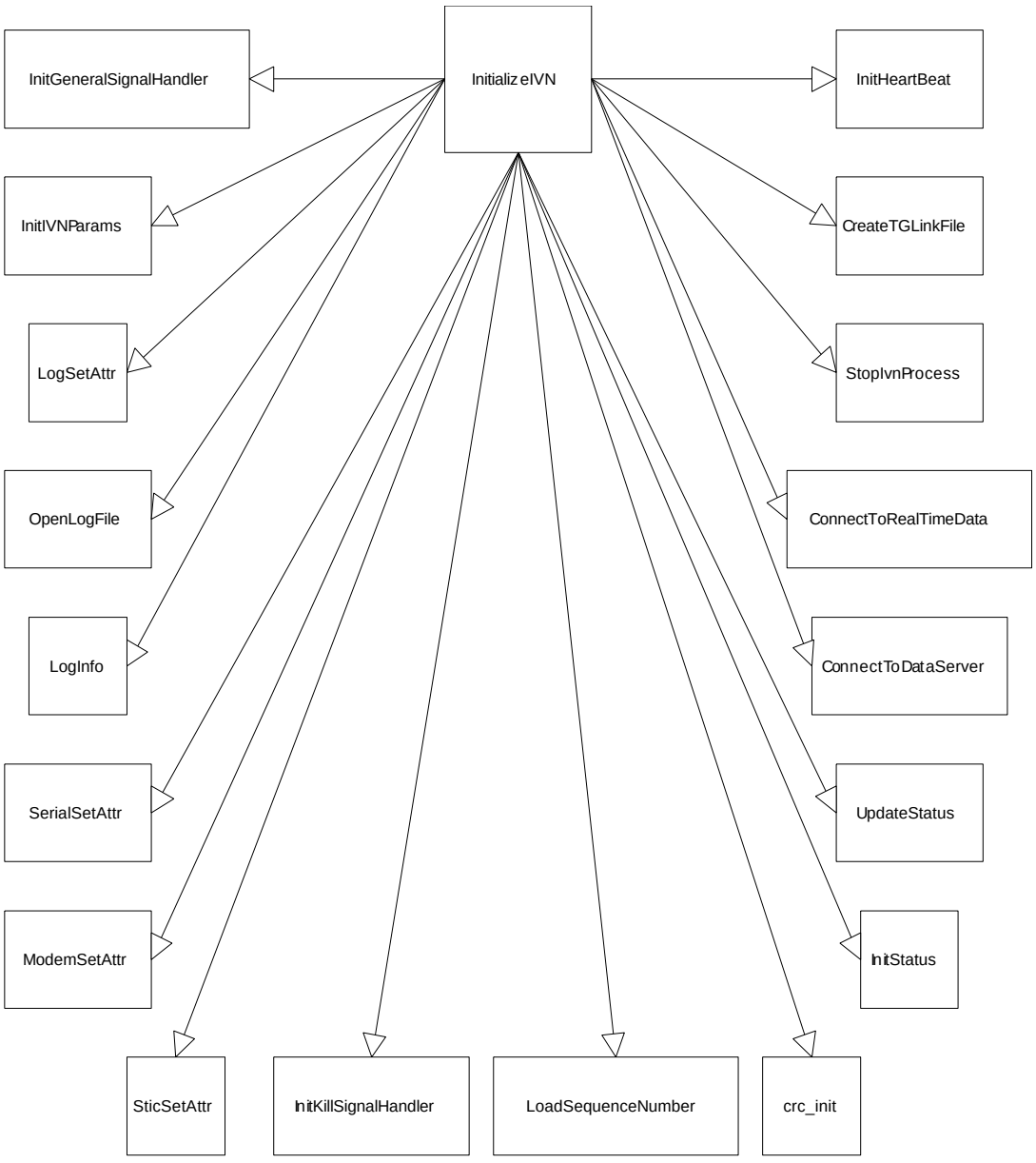

Figure 9. IVN Initialization

Function	Description
ConnectToDataServer	Establishes the socket connection to the Data Server process.
ConnectToRealTimeData	Establishes the socket connection to the Real Time Data process and refreshes the real time data.
crc_init	Initializes the CRC tables.
CreateTGLinkFile	Creates a text file on disk which contains the IDs and locations of each TransGuide link.
InitGeneralSignalHandler	Assigns the handler for catching unassigned signals.
InitHeartBeat	Initializes the heartbeat timer and assigns the heartbeat timer handler.
InitIVNParams	Loads and assigns the values of the IVN configurable parameters from disk file.
InitKillSignalHandler	Assigns the function to catch and handle the SIGTERM signal.
InitStatus	Creates and initializes the shared memory used to communicate to the status GUI.
LoadSequenceNumber	Loads and assigns the last used STM sequence number from disk file.
LogInfo	Logs informational messages to the status log and/or standard output.
LogSetAttr	Sets the message logging attributes.
ModemSetAttr	Sets the modem control function attributes.
OpenLogFile	Opens the status log file.
SerialSetAttr	Sets the serial I/O function attributes.
SticSetAttr	Sets the STIC communication attributes.
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.
UpdateStatus	Updates the status GUI shared memory.

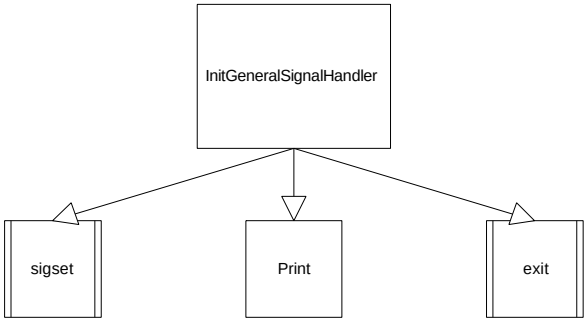

Figure 10. InitializeGeneralSignalHandler

Function	Description
exit	C library function.
Print	Prints message to standard output.
sigset	C library function.

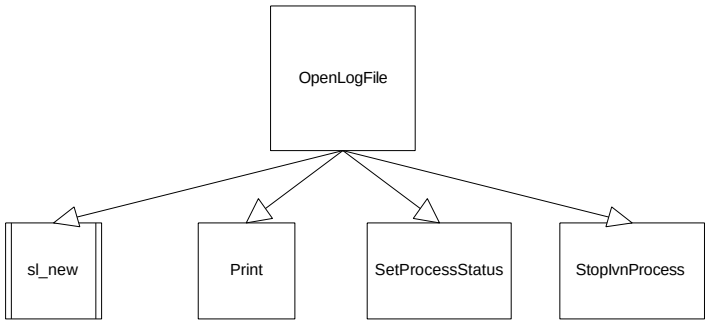

Figure 11. Open Log File

Function	Description
Print	Prints message to standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
sl_new	Status Logger library function. Creates a new instance of a status log file.
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

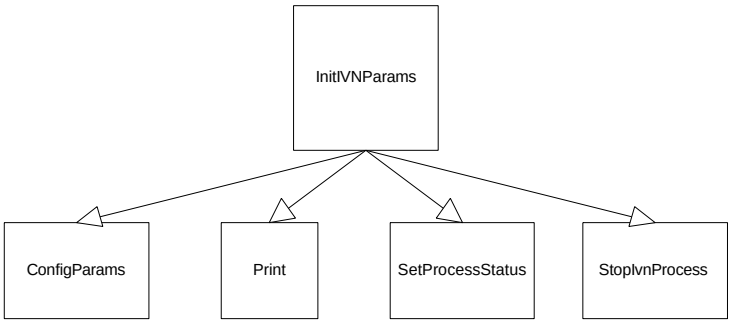

Figure 12. Initialize Parameters

Function	Description
ConfigParams	Loads and assigns parameters from disk file.
Print	Prints message to standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

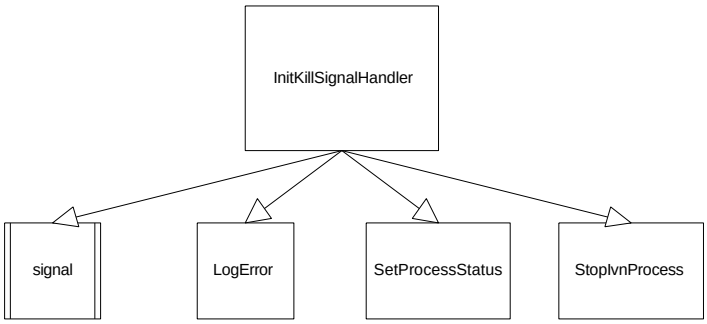

Figure 13. InitializeKillSignalHandler

Function	Description
LogError	Logs error messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
signal	C library function.
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

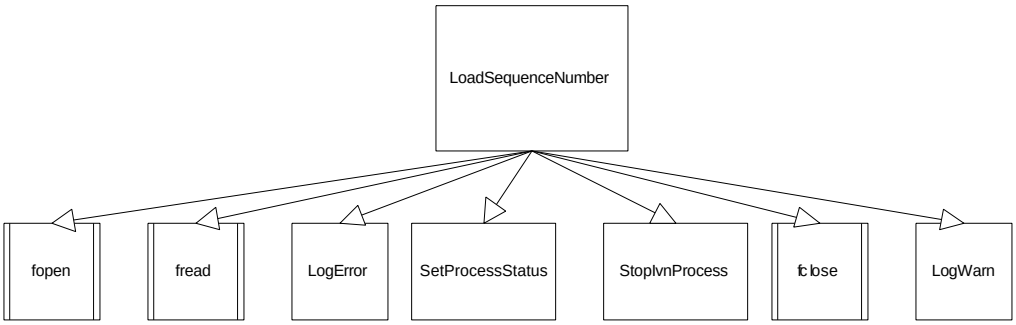

Figure 14. Load Sequence Number

Function	Description
fclose	C library function.
fopen	C library function.
fread	C library function.
LogError	Logs error messages to the status log and/or standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

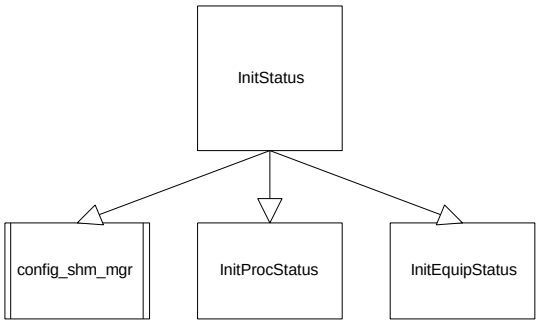

Figure 15. Initialize Status GUI Shared Memory

Function	Description
config_shm_mgr	Shared Memory Management library function. Initializes the shared memory manager library.
InitEquipStatus	Creates and initializes equipment status shared memory.
InitProcStatus	Creates and initializes process status shared memory.

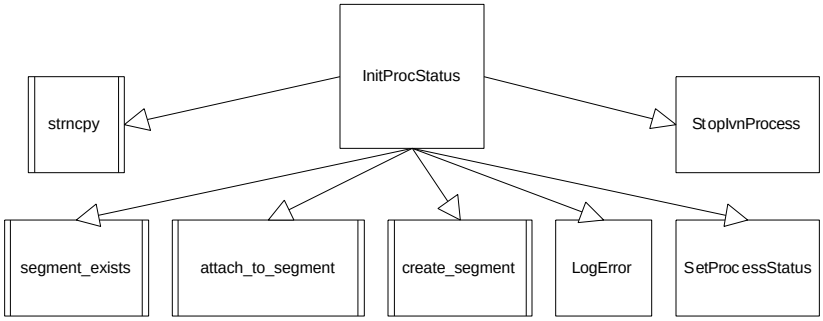

Figure 16. Initialize Process Status Shared Memory

Function	Description
attach_to_segment	Shared Memory Management library function. Attaches to an existing shared memory segment.
create_segment	Shared Memory Management library function. Creates a new shared memory segment.
LogError	Logs error messages to the status log and/or standard output.
segment_exists	Shared Memory Management library function. Returns true if the specified shared memory segment exists.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.
strncpy	C library function.

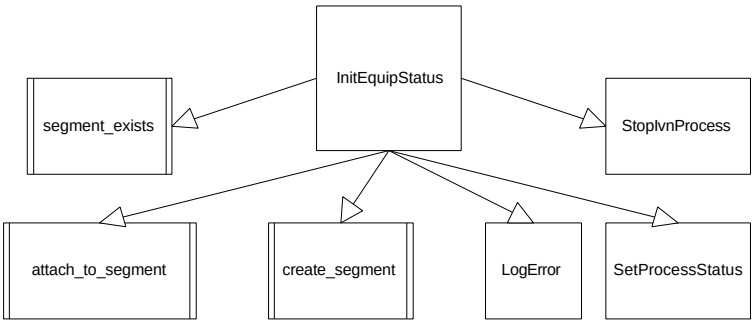

Figure 17. Initialize Equipment Status

Function	Description
attach_to_segment	Shared Memory Management library function. Attaches to an existing shared memory segment.
create_segment	Shared Memory Management library function. Creates a new shared memory segment.
LogError	Logs error messages to the status log and/or standard output.
segment_exists	Shared Memory Management library function. Returns true if the specified shared memory segment exists.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

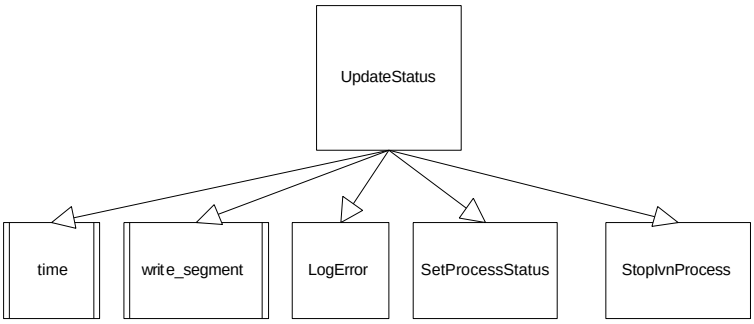

Figure 18. Update Status

Function	Description
LogError	Logs error messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.
time	C library function.
write_segment	Shared Memory Management library function. Writes data to the specified shared memory segment.

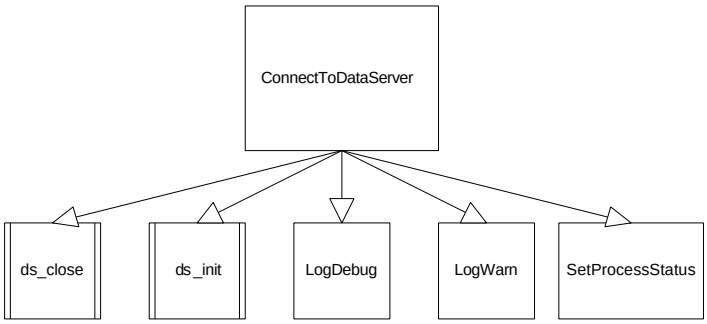

Figure 19. Connect to Data Server

Function	Description
ds_close	Data Server library function. Closes the connection to the Data Server.
ds_init	Data Server library function. Establishes the connection to the Data Server.
LogDebug	Logs debug messages to the status log and/or standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).

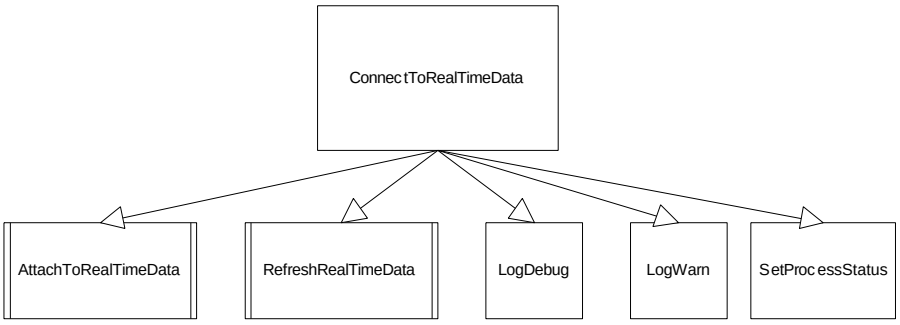

Figure 20. Connect To RealTime Data

Function	Description
AttachToRealTimeData	Real Time Data library function. Establishes the connection to the real time data stream.
LogDebug	Logs debug messages to the status log and/or standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
RefreshRealTimeData	Real Time Data library function. Refreshes the link real time data with the current values.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).

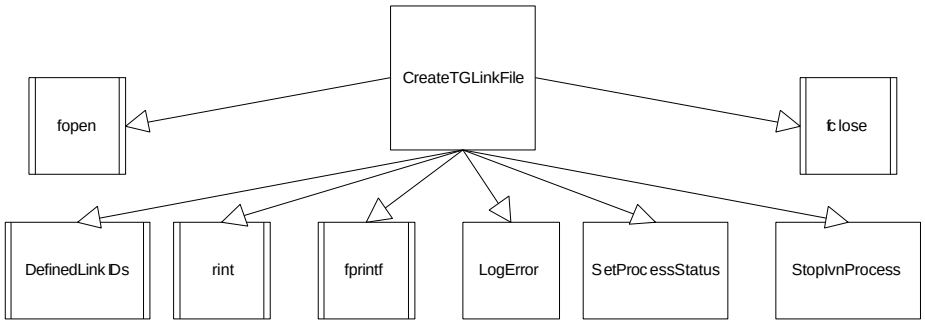

Figure 21. Create TransGuide Link ID File

Function	Description
DefinedLinkIDs	Real Time Data library function. Returns a pointer to the list of TransGuide link IDs.
fclose	C library function.
fopen	C library function.
fprintf	C library function.
LogError	Logs error messages to the status log and/or standard output.
rint	C library function.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

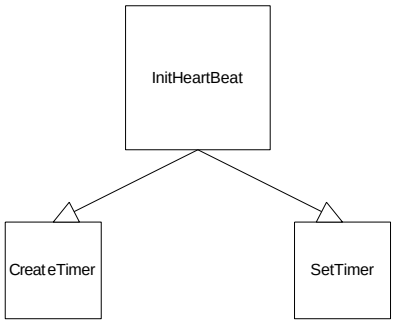

Figure 22. Initialize Heartbeat Timer

Function	Description
CreateTimer	Creates a timer and assigns a handler for the timer.
SetTimer	Sets timer expiration and interval.

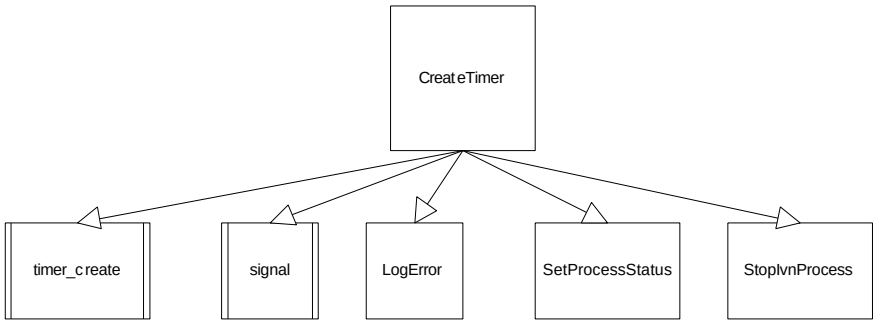

Figure 23. Create Timer

Function	Description
LogError	Logs error messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
signal	C library function.
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.
timer_create	C library function.

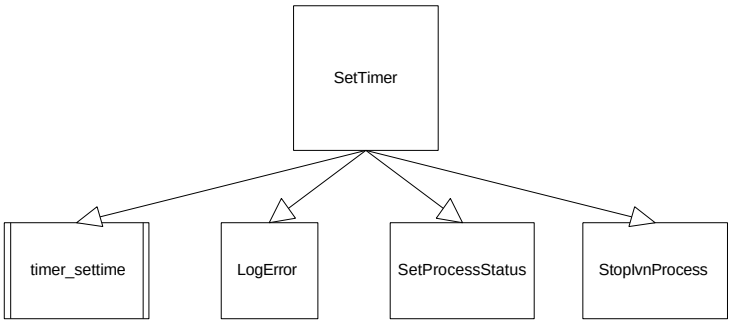

Figure 24. Set Timer

Function	Description
LogError	Logs error messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.
Timer_settime	C library function.

4.2.3.2 Initialization of Data Structures

Upon completion of initialization and prior to execution of the main loop, the IVN process allocates and initializes the data structures required for maintaining the real time data, formatting it per the IVN protocol and transmitting it to the STIC system. The four data structures required for this include: a local copy of the real time TransGuide link ID array, a local copy of the TransGuide incident array, the link speed STIC Transmission Message (STM) array, and the incident STM array.

The local copy of the link ID array is a duplicate of the TransGuide link ID array with an additional field included with each link entry. This field is the link location computed according to the IVN communication protocol. To reduce the data required for a location reference, the protocol computes link locations as relative values from an origin location. If a link is not within the required range from the origin to be defined as a relative location, the absolute location of the link is recorded. The location references for the links (relative or absolute) must be determined before the link data can be formatted per the IVN communication protocol. The local link ID array includes the TransGuide link ID data and the computed location reference for each link. The functions which create the local link data array are described below.

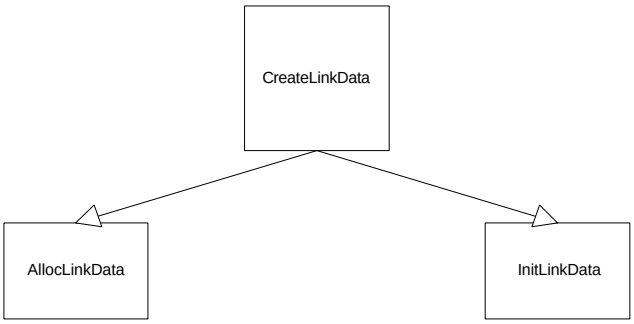

Figure 25. Create Link Speed Data

Function	Description
AllocLinkData	Allocates memory for the local copy of the TransGuide link data array.
InitLinkData	Initializes the local copy of the TransGuide link data array with the current link data and determines and computes the link location reference (either local or global).

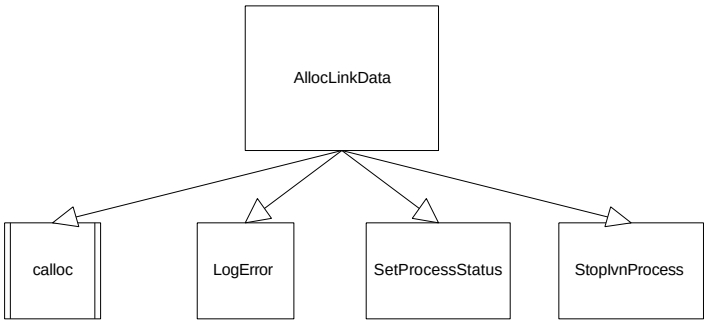

Figure 26. Allocate Link Speed Data Array

Function	Description
calloc	C library function.
LogError	Logs error messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

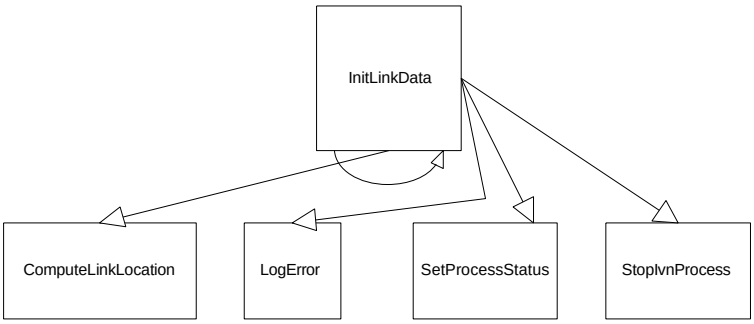

Figure 27. Initialize Link Speed Data Array

Function	Description
ComputeLinkLocation	Determines the link's location reference (either local or global) and calculates either the locally or globally referenced latitude and longitude.
LogError	Logs error messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

The local copy of the incident array is a duplicate of the TransGuide incident array with an additional field included with each incident entry. This field is the incident location computed according to the IVN communication protocol. To reduce the data required for a location reference, the protocol computes incident locations as relative values from an origin location. If an incident is not within the required range from the origin to be defined as a relative location, the absolute location of the incident is recorded. The location references for the incidents (relative or absolute) must be determined before the incident data can be formatted per the IVN communication protocol. The local incident array includes the TransGuide incident data and the computed location reference for each incident. The functions which create the local incident data array are described below.

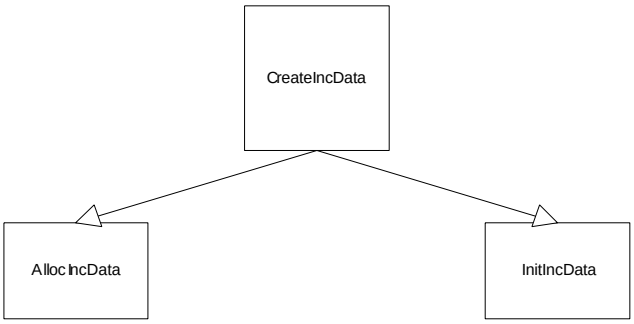

Figure 28. Create Incident Data

Function	Description
AllocIncData	Allocates memory for the local copy of the TransGuide incident data array.
InitIncData	Initializes the local copy of the TransGuide incident data array with the current incident data and determines and computes the incident location reference (either local or global).

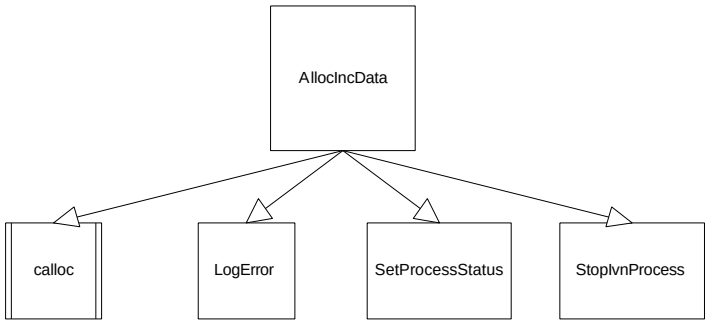

Figure 29. Allocate Incident Data Array

Function	Description
Calloc	C library function.
LogError	Logs error messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

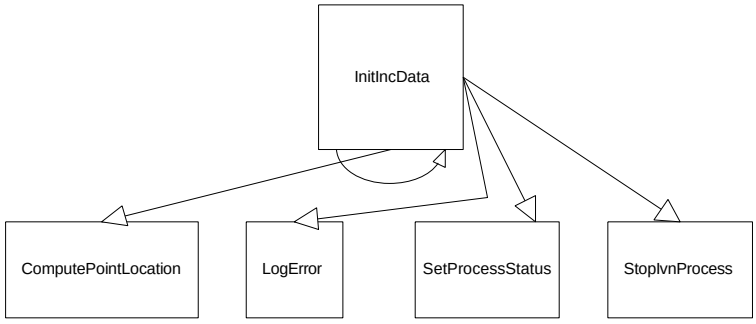

Figure 30. Initialize Incident Data Array

Function	Description
ComputePointLocation	Determines the point's location reference (either local or global) and calculates either the locally or globally referenced latitude and longitude.
LogError	Logs error messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

The link speed STM array is an array of data structures (STMs) which contain the link speed data formatted per the IVN communication protocol. Each STM contains the information for a single Traffic Information Message (TIM). Each TIM contains the link speed data for a number of TransGuide link IDs. The details of the format of the STM and TIM are described in the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol*. The link speed STM array is created at startup, and is updated to reflect changes in link speed when the TransGuide real time link speed data is updated. The functions which create the link speed STM array are described below.

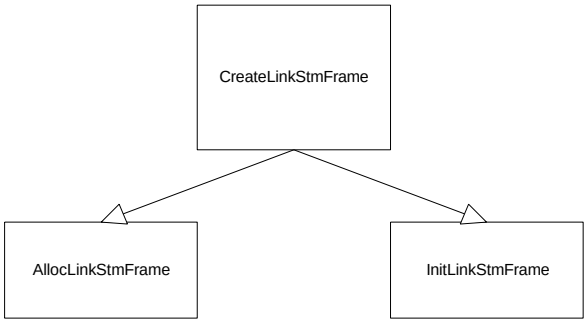

Figure 31. Create Link Speed STM Array

Function	Description
AllocLinkStmFrame	Allocates memory for the link speed STM data array.
InitLinkStmFrame	Initializes the values of the fields of each STM in the link speed STM data array as defined by the IVN communication protocol.

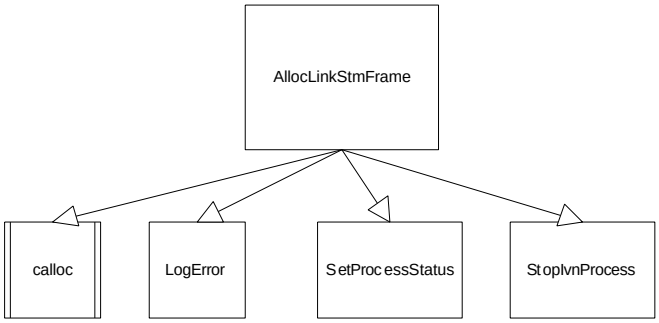

Figure 32. Allocate Link Speed STM Array

Function	Description
calloc	C library function.
LogError	Logs error messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

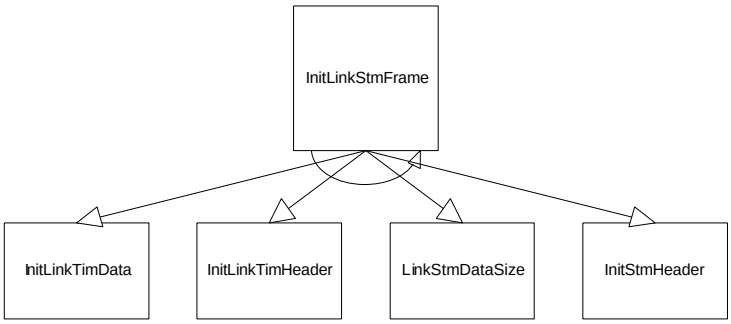

Figure 33. Initialize Link Speed STM Array

Function	Description
InitLinkTimData	Initializes the data fields of a link speed TIM as defined by the IVN communication protocol.
InitLinkTimHeader	Initializes the header fields of a link speed TIM as defined by the IVN communication protocol.
InitStmHeader	Initializes the header data fields in a link speed or incident STM as defined by the IVN communication protocol.
LinkStmDataSize	Returns the size of the link STM data in bytes.

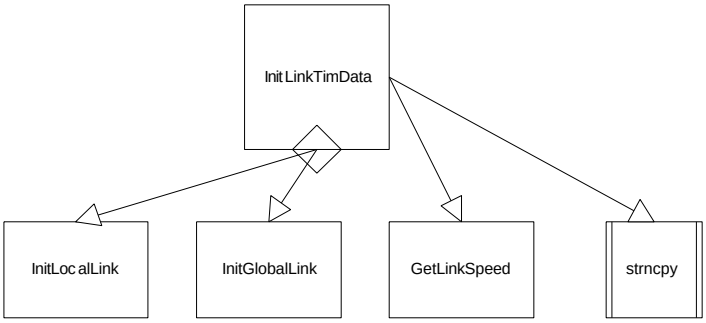

Figure 34. Initialize Link Speed TIM Data

Function	Description
GetLinkSpeed	Returns the current speed for the specified TransGuide link ID.
InitGlobalLink	Initializes the fields in a globally referenced link location as defined by the IVN communication protocol.
InitLocalLink	Initializes the fields in a locally referenced link location as defined by the IVN communication protocol.
strncpy	C library function.

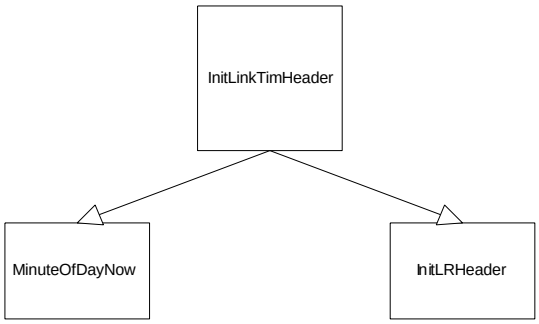

Figure 35. Initialize Link Speed TIM Header

Function	Description
InitLRHeader	Initializes the fields in a location reference header as defined by the IVN communication protocol.
MinuteOfDayNow	Returns the current minute of the day.

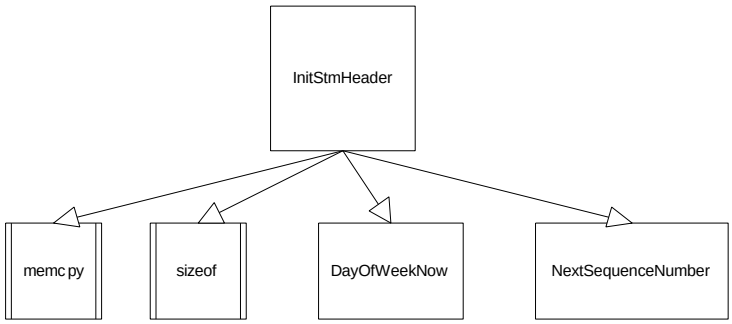

Figure 36. Initialize STM Header

Function	Description
DayOfWeekNow	Returns the current day of week.
memcpy	C library function.
NextSequenceNumber	Returns next available STM sequence number.
sizeof	C library function.

The incident STM array is an array of data structures (STMs) which contains the incident data formatted per the IVN communication protocol. Each STM contains the information for a single Traffic Information Message (TIM). Each TIM contains the incident data for a number of TransGuide incidents. The details of the format of the STM and TIM are described in the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol*. The incident STM array is created at startup, and is updated to reflect changes in incidents when the TransGuide real time incident data is updated. The functions which create the incident STM array are described below.

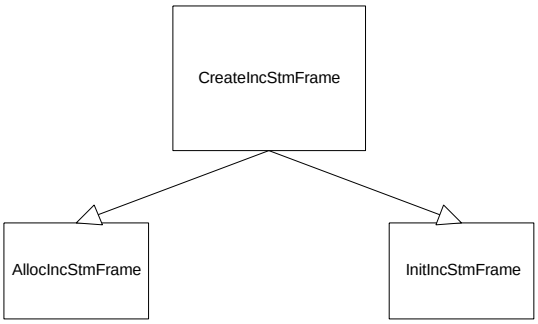

Figure 37. Create Incident STM Array

Function	Description
AllocIncStmFrame	Allocates memory for the incident STM data array.
InitIncStmFrame	Initializes the values of the fields in the incident STM data array as defined by the IVN communication protocol.

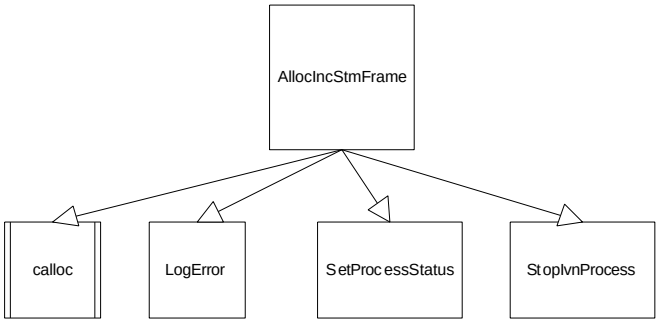

Figure 38. Allocate Incident STM Array

Function	Description
calloc	C library function.
LogError	Logs error messages to the status log and/or standard output.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

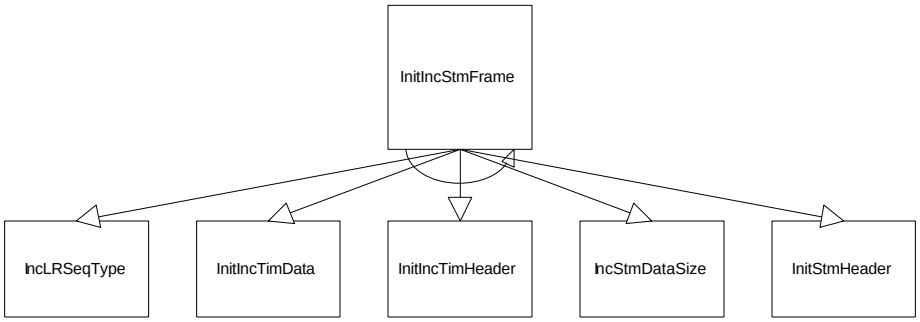

Figure 39. Initialize Incident STM Array

Function	Description
IncLRSeqType	Returns the location reference type of the incident location.
IncStmDataSize	Returns the size of the incident STM data in bytes.
InitIncTimData	Initializes the data fields of an incident TIM as defined by the IVN communication protocol.
InitIncTimHeader	Initializes the header fields of an incident TIM as defined by the IVN communication protocol.
InitStmHeader	Initializes the header data fields in a link speed or incident STM as defined by the IVN communication protocol.

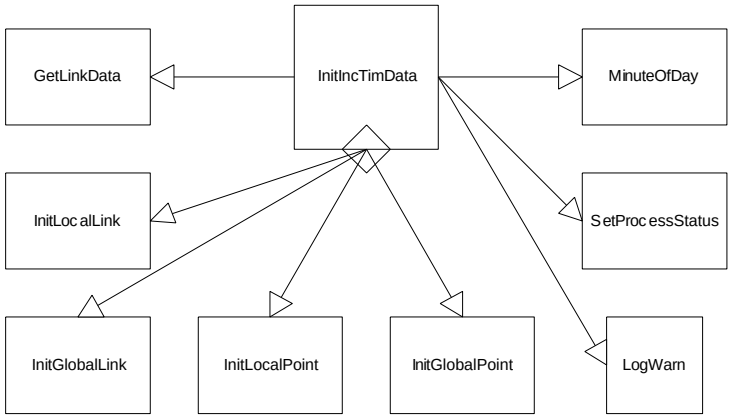

Figure 40. Initialize Incident TIM Data

Function	Description
GetLinkData	Returns the link data for the specified TransGuide link ID.
InitGlobalLink	Initializes the fields in a globally referenced link location as defined by the IVN communication protocol.
InitGlobalPoint	Initializes the fields in a globally referenced point location as defined by the IVN communication protocol.
InitLocalLink	Initializes the fields in a locally referenced link location as defined by the IVN communication protocol.
InitLocalPoint	Initializes the fields in a locally referenced point location as defined by the IVN communication protocol.
LogWarn	Logs warning messages to the status log and/or standard output.
MinuteOfDay	Returns current minute of the day.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).

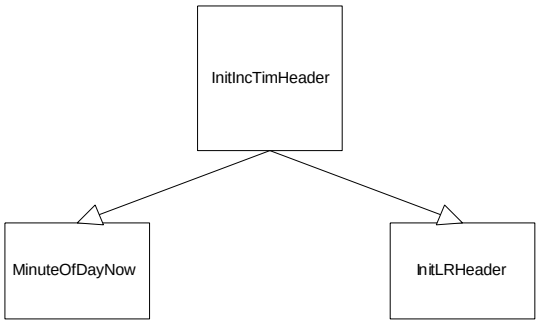

Figure 41. Initialize Incident TIM Header

Function	Description
InitLRHeader	Initializes the fields in a location reference header as defined by the IVN communication protocol.
MinuteOfDayNow	Returns the current minute of the day.

4.2.3.3 Modem Control

The IVN process communicates to the STIC message encoder via dial-up modem. The connection between the IVN modem and the STIC modem is checked on each iteration of the main loop and is re-established if the connection is down. The functions which control the operation of the modem are described below.

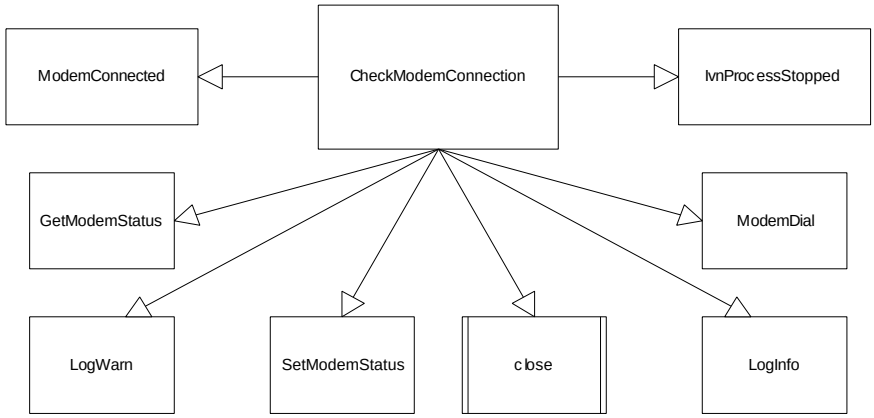

Figure 42. Check Modem Connection

Function	Description
close	C library function.
GetModemStatus	Returns modem equipment status.
IvnProcessStopped	Returns true if the IVN process has been stopped.
LogInfo	Logs informational messages to the status log and/or standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
ModemConnected	Returns true if the IVN modem is connected to a remote modem.
ModemDial	Issues modem commands to dial and connect to a remote modem and waits for the connection to complete.
SetModemStatus	Sets the modem equipment status.

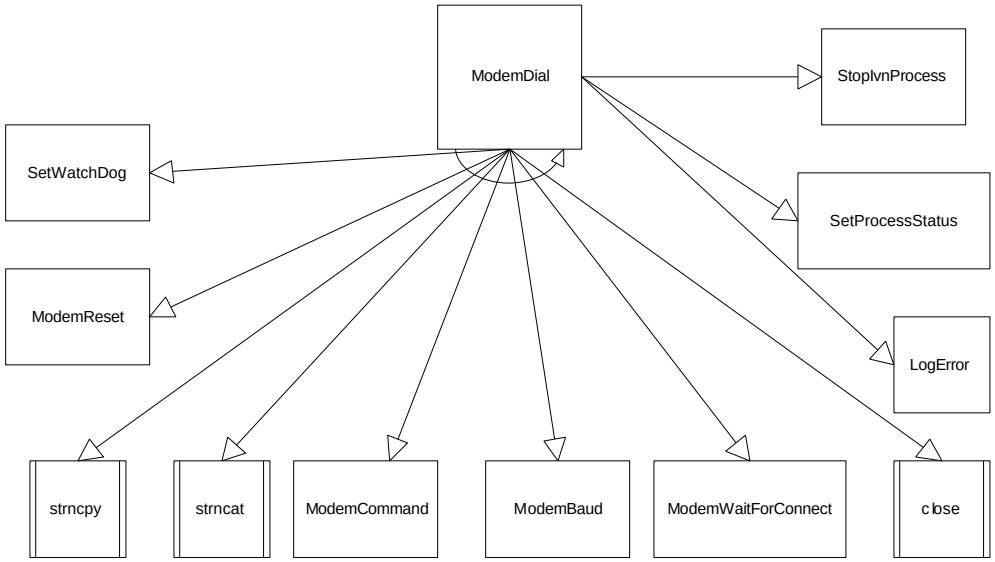

Figure 43. DialModem

Function	Description
close	C library function.
LogError	Logs error messages to the status log and/or standard output.
ModemBaud	Returns the modem baud rate command for the specified baud rate.
ModemCommand	Issues a command to the modem and reads and verifies modem reply.
ModemReset	Opens modem serial port, resets modem and returns file descriptor for serial port.
ModemWaitForConnect	Waits for modem to connect to remote modem.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
SetWatchDog	Sets the IVN process watchdog timer flag.
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.
strncat	C library function.
strncpy	C library function.

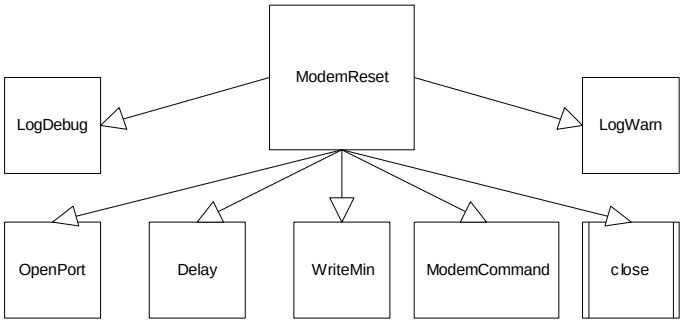

Figure 44. ResetModem

Function	Description
close	C library function.
Delay	Delays for a specified number of seconds.
LogDebug	Logs debug messages to the status log and/or standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
ModemCommand	Issues a command to the modem and reads and verifies modem reply.
OpenPort	Opens a serial port and configures the serial communication parameters.
WriteMin	Writes a minimum number of bytes of data to a serial port.

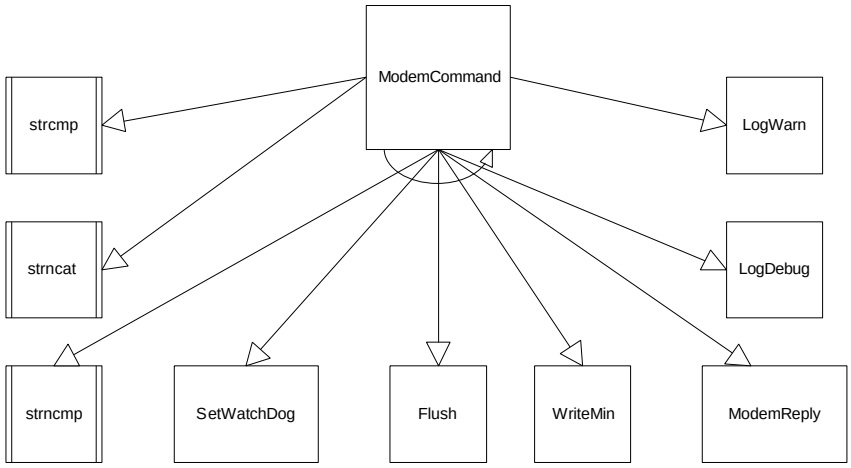

Figure 45. CommandModem

Function	Description
Flush	Flushes a serial port.
LogDebug	Logs debug messages to the status log and/or standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
ModemReply	Reads and verifies a reply to a modem command.
SetWatchDog	Sets the IVN process watchdog timer flag.
strcmp	C library function.
strncat	C library function.
strncmp	C library function.
WriteMin	Writes a minimum number of bytes of data to a serial port.

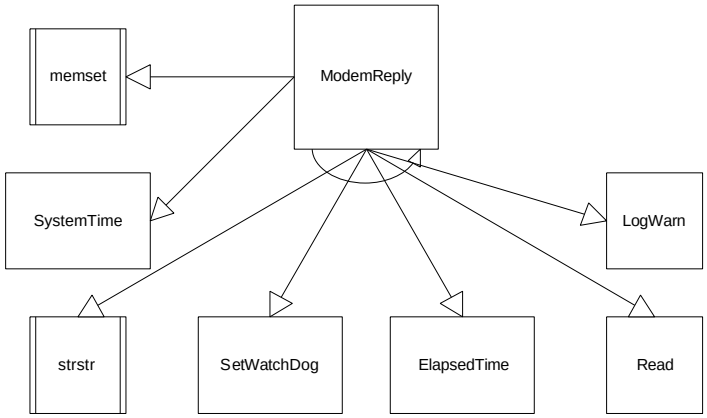

Figure 46. Read Modem Reply

Function	Description
ElapsedTime	Returns the elapsed time in seconds from a specified start time.
LogWarn	Logs warning messages to the status log and/or standard output.
memset	C library function.
Read	Reads data from a serial port.
SetWatchDog	Sets the IVN process watchdog timer flag.
strstr	C library function.
SystemTime	Returns the current system time.

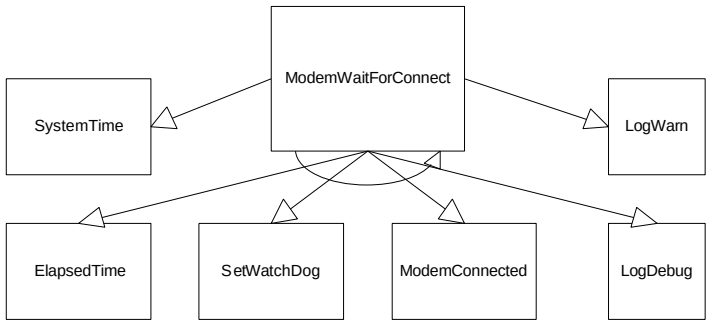

Figure 47. Wait for Modem Connection

Function	Description
ElapsedTime	Returns the elapsed time in seconds from a specified start time.
LogDebug	Logs debug messages to the status log and/or standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
ModemConnected	Returns true if the IVN modem is connected to a remote modem.
SetWatchDog	Sets the IVN process watchdog timer flag.
SystemTime	Returns the current system time.

4.2.3.4 RealTime Data Update

On each iteration of the main loop the TransGuide real time data is refreshed with the current values, and the STM data arrays are updated to reflect any changes. For the link speed STM array, the update only involves modifying the speed fields in the TIMs for any links which had a change in speed. For the incident STM array, the update involves deleting from the TIMs incidents which no longer exist and adding to the TIMs any new incidents. The functions which perform the update of the real time data are described below.

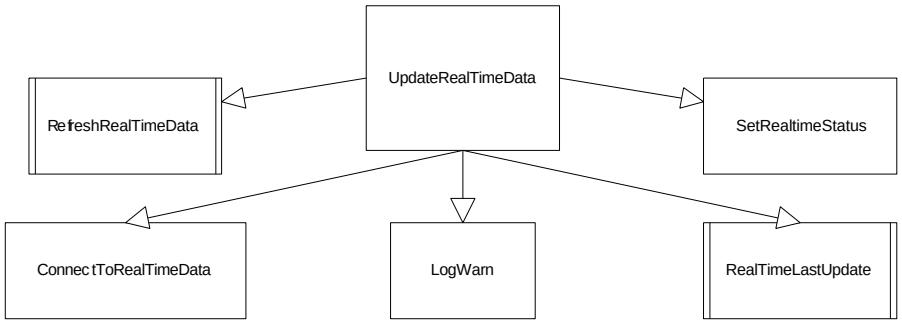

Figure 48. Update RealTime Data

Function	Description
ConnectToRealTimeData	Establishes the socket connection to the Real Time Data process (if necessary) and refreshes the real time data.
LogWarn	Logs warning messages to the status log and/or standard output.
RealTimeLastUpdate	Real Time Data library function. Returns the last update time for the specified type of real time data.
RefreshRealTimeData	Real Time Data library function. Refreshes the link real time data with the current values.
SetRealtimeStatus	Updates the real time data equipment status.

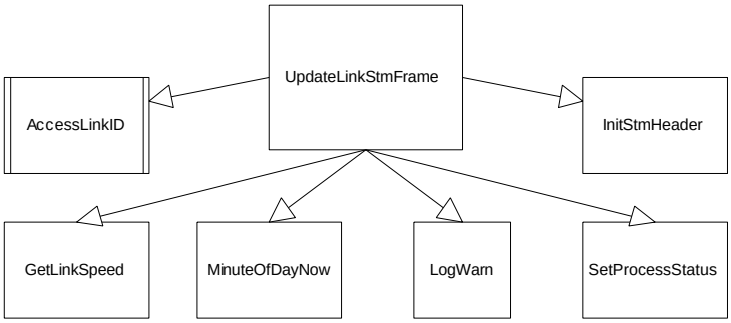

Figure 49. Update Link Speed STM Array

Function	Description
AccessLinkID	Real Time Data library function. Returns the data for the specified TransGuide link ID.
GetLinkSpeed	Returns the current speed for the specified TransGuide link ID.
InitStmHeader	Initializes the header data fields in a link speed or incident STM as defined by the IVN communication protocol.
LogWarn	Logs warning messages to the status log and/or standard output.
MinuteOfDayNow	Returns the current minute of the day.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).

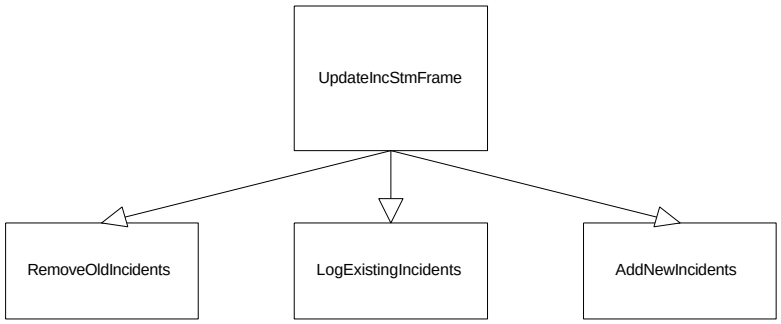

Figure 50. Update Incident STM Array

Function	Description
AddNewIncidents	Adds new incidents to the incident STM data array.
LogExistingIncidents	Identifies incidents which already exist in the incident STM array.
RemoveOldIncidents	Removes incidents which no longer exist from the incident STM array.

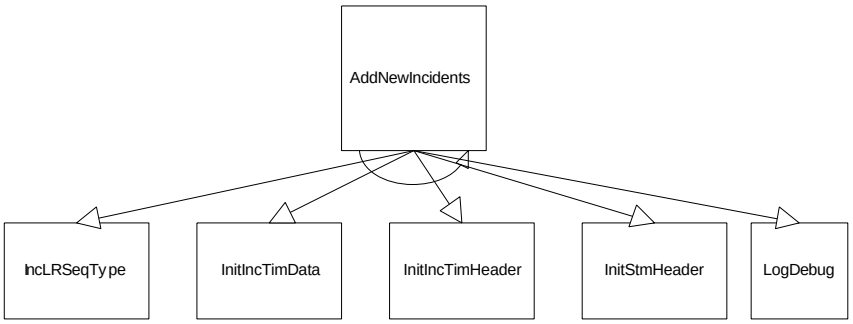

Figure 51. Add New Incidents

Function	Description
IncLRSeqType	Returns the location reference type of the incident location.
InitIncTimData	Initializes the data fields of an incident TIM as defined by the IVN communication protocol.
InitIncTimHeader	Initializes the header fields of an incident TIM as defined by the IVN communication protocol.
InitStmHeader	Initializes the header data fields in a link speed or incident STM as defined by the IVN communication protocol.
LogDebug	Logs debug messages to the status log and/or standard output.

4.2.3.5 STIC Message Transmission

On each iteration of the main loop after the real time data has been updated, the modified STM arrays are transmitted to the STIC message encoder via the modem. The main steps required to transmit the STMs to the STIC include: packing the STM data arrays to remove white space, formatting the STM data into transmission packets per the STIC communication protocol, and transmitting the packets to the STIC via the modem. The functions which transmit the STM data arrays to the STIC message encoder are described below.

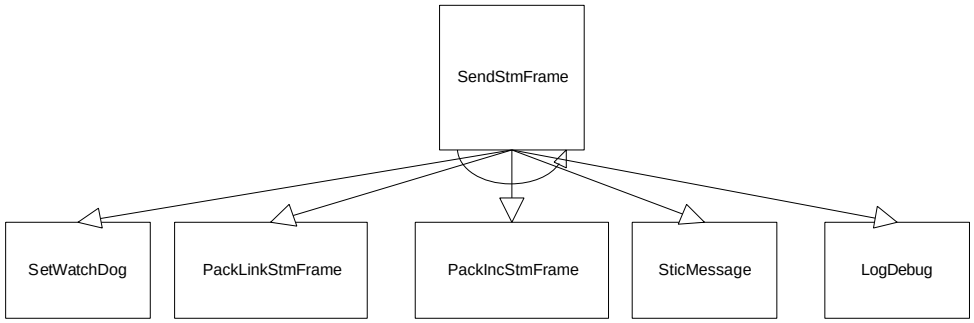

Figure 52. Send STM Array

Function	Description
LogDebug	Logs debug messages to the status log and/or standard output.
PackIncStmFrame	Invokes the incident STM packing function for each STM in the incident STM data array.
PackLinkStmFrame	Invokes the link speed STM packing function for each STM in the link speed STM data array.
SetWatchDog	Sets the IVN process watchdog timer flag.
SticMessage	Issues a message to the STIC system and reads and verifies the STIC response.

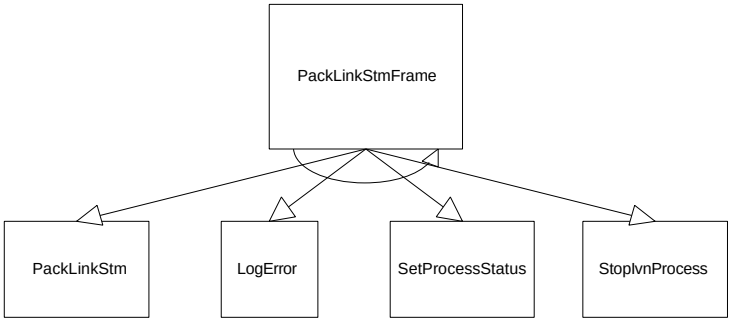

Figure 53. Pack Link Speed STM Array

Function	Description
LogError	Logs error messages to the status log and/or standard output.
PackLinkStm	Packs a link speed STM, stores the packed data to the STIC transmission buffer, computes and appends the CRC to the packed data, and performs the zero byte insertion on the packed data.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

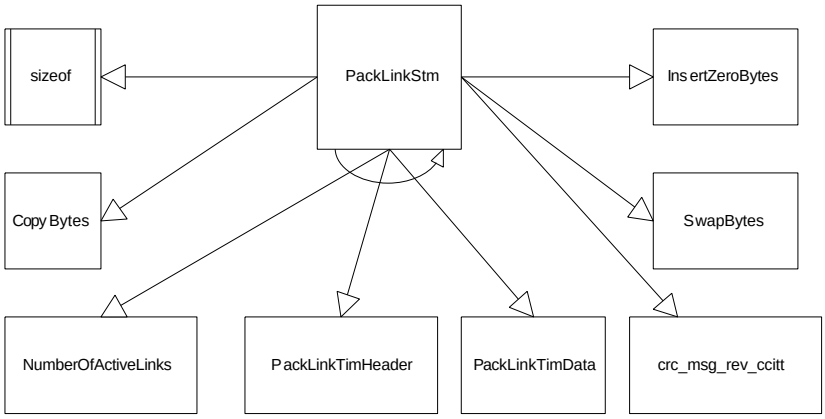

Figure 54. Pack Link Speed STM

Function	Description
CopyBytes	Copies specified number of bytes from source to destination buffer, checking for overflow.
crc_msg_rev_ccitt	Calculates the reverse CCITT crc on a data buffer.
InsertZeroBytes	Inserts a zero (0) byte after the occurrence of the first two bytes of the start of message flag (SOM) in the data field of an STM to eliminate false SOMs.
NumberOfActiveLinks	Returns the number of active links in a link speed TIM.
PackLinkTimData	Packs the data fields in a link speed TIM and copies the packed data to the STIC transmission buffer.
PackLinkTimHeader	Packs the fields in the header of a link speed TIM and copies the packed data to the STIC transmission buffer.
sizeof	C library function.
SwapBytes	Swaps the bytes in word.

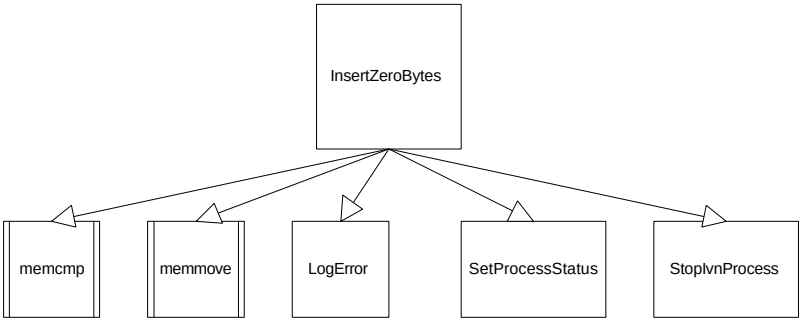

Figure 55. InsertZeroBytes

Function	Description
LogError	Logs error messages to the status log and/or standard output.
memcmp	C library function.
memmove	C library function.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

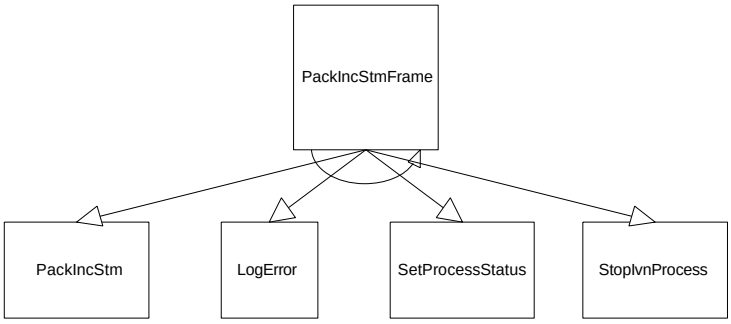

Figure 56. Pack Incident STM Array

Function	Description
LogError	Logs error messages to the status log and/or standard output.
PackIncStm	Packs an incident STM, stores the packed data to the STIC transmission buffer, computes and appends the CRC to the packed data, and performs the zero byte insertion on the packed data.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

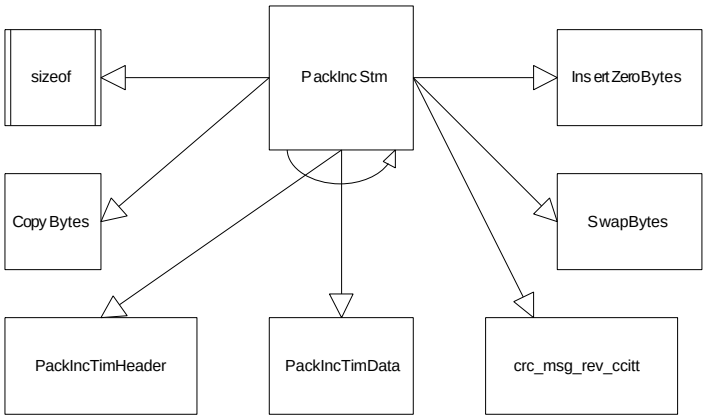

Figure 57. Pack Incident STM

Function	Description
CopyBytes	Copies specified number of bytes from source to destination buffer, checking for overflow.
crc_msg_rev_ccitt	Calculates the reverse CCITT crc on a data buffer.
InsertZeroBytes	Inserts a zero (0) byte after the occurrence of the first two bytes of the start of message flag (SOM) in the data field of an STM to eliminate false SOMs.
PackIncTimData	Packs the data fields in an incident TIM and copies the packed data to the STIC transmission buffer.
PackIncTimHeader	Packs the fields in the header of an incident TIM and copies the packed data to the STIC transmission buffer.
sizeof	C library function.
SwapBytes	Swaps the bytes in a word.

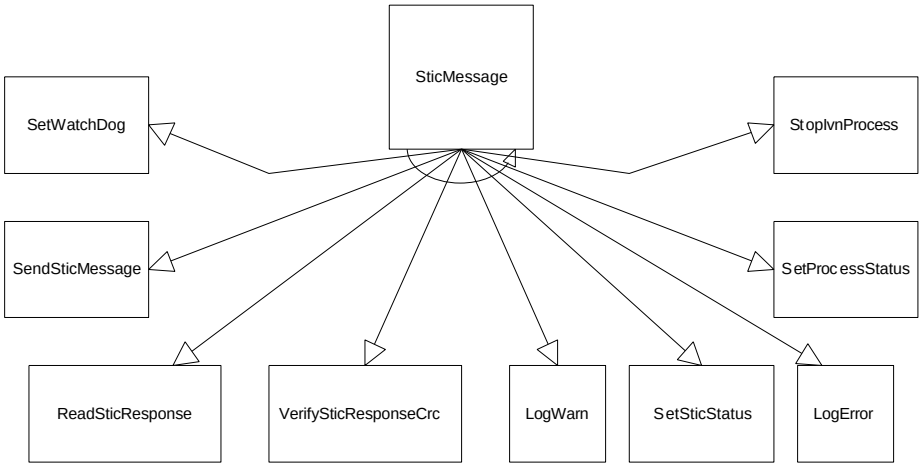

Figure 58. STIC Message

Function	Description
LogError	Logs error messages to the status log and/or standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
ReadSticResponse	Reads a response message from the STIC system.
SendSticMessage	Formats and transmits a message to the STIC system.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
SetSticStatus	Sets the STIC equipment status.
SetWatchDog	Sets the IVN process watchdog timer flag.
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.
VerifySticResponseCrc	Validates the CRC of the STIC response message.

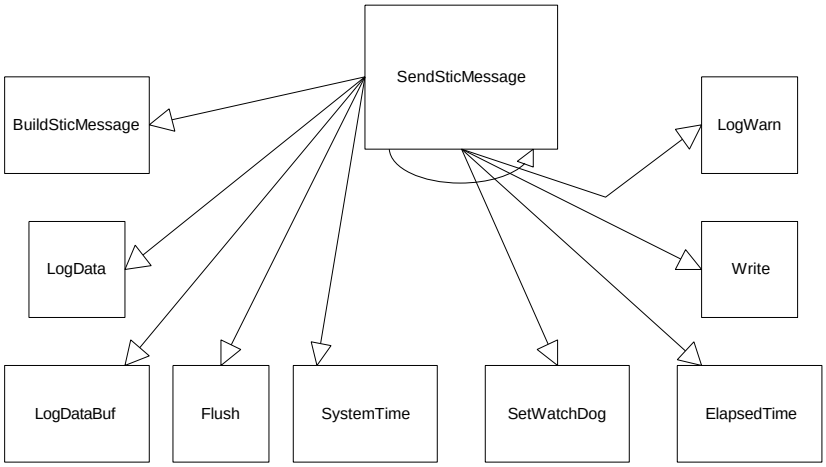

Figure 59. Send STIC Message

Function	Description
BuildSticMessage	Builds a message formatted according to the STIC communication protocol.
ElapsedTime	Returns the elapsed time in seconds from a specified start time.
Flush	Flushes a serial port.
LogData	Logs process data to the status log and/or the standard output.
LogDataBuf	Logs a data buffer to the status log and/or the standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
SetWatchDog	Sets the IVN process watchdog timer flag.
SystemTime	Returns the current system time.
Write	Writes data to a serial port.

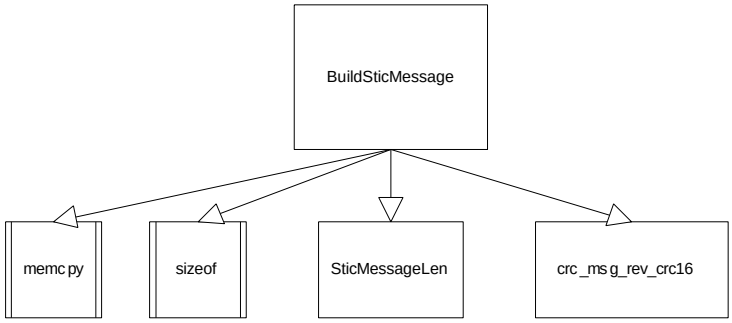

Figure 60. Build STIC Message

Function	Description
Crc_msg_rev_crc16	Calculates the reverse crc 16 crc on a data buffer.
memcpy	C library function.
Sizeof	C library function.
SticMessageLen	Returns the length of a specified STIC message type.

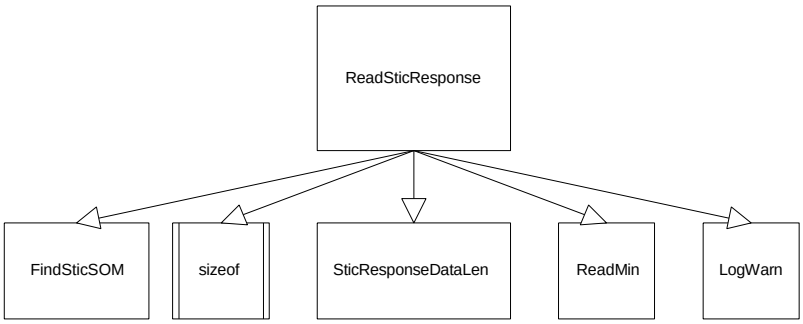

Figure 61. Read STIC Response

Function	Description
FindSticSOM	Detects the start of message flag in a STIC response.
LogWarn	Logs warning messages to the status log and/or standard output.
ReadMin	Reads a minimum number of bytes of data from a serial port.
sizeof	C library function.
SticResponseDataLen	Returns the data length for the specified STIC response message.

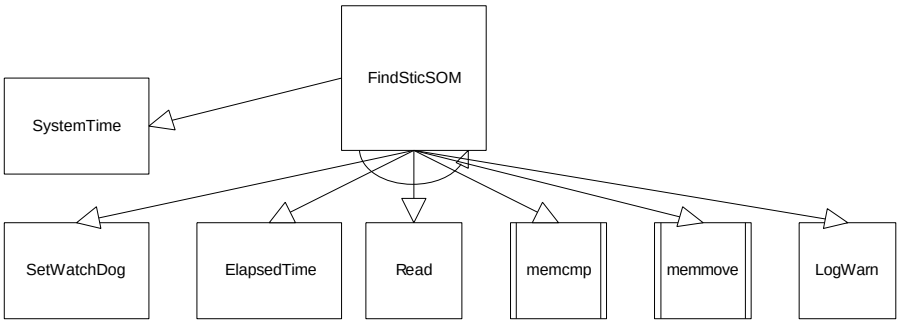

Figure 62. Find STIC Start of Message

Function	Description
ElapsedTime	Returns the elapsed time in seconds from a specified start time.
LogWarn	Logs warning messages to the status log and/or standard output.
memcmp	C library function.
memmove	C library function.
Read	Reads data from a serial port.
SetWatchDog	Sets the IVN process watchdog timer flag.
SystemTime	Returns the current system time.

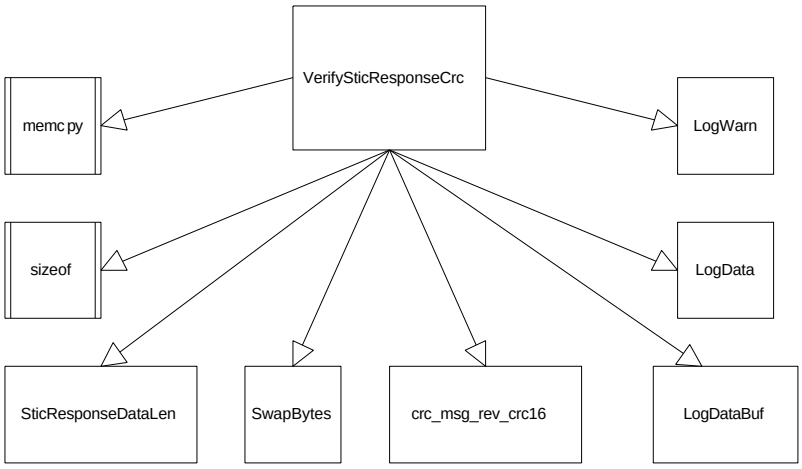

Figure 63. Verify STIC Response CRC

Function	Description
crc_msg_rev_crc16	Calculates the reverse crc 16 crc on a data buffer.
LogData	Logs process data to the status log and/or the standard output.
LogDataBuf	Logs a data buffer to the status log and/or the standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
memcpy	C library function.
sizeof	C library function.
SticResponseDataLen	Returns the data length for the specified STIC response message.
SwapBytes	Swaps the bytes in word.

4.2.3.6 ShutDown

Upon exiting the main loop, due either to receipt of a termination signal or the occurrence of a fatal error, the IVN process shutdown procedure is executed. The shutdown procedure updates the process status, closes communication interfaces, and exits the process. The functions which perform the shutdown are described below.

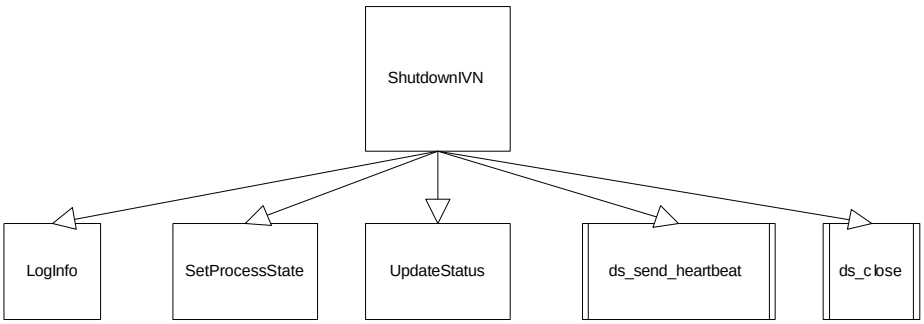

Figure 64. ShutDown IVN

Function	Description
ds_close	Data Server library function. Closes the connection to the Data Server.
ds_send_heartbeat	Data Server library function. Sends the process status to the Data Server.
LogInfo	Logs informational messages to the status log and/or standard output.
SetProcessState	Sets the process state field of the process status structure.
UpdateStatus	Updates the status GUI shared memory.

4.2.3.7 SignalHandlers

The IVN signal handlers capture and handle asynchronous software signals. The IVN process includes three signal handlers. A general signal handler captures unassigned signals. Depending on the signal received, the unassigned signals are either ignored or cause the process to be shut down on fatal error. The kill signal handler captures the process termination signal and causes the IVN process to gracefully shut down. The heartbeat timer handler captures the signal generated by the heartbeat timer and issues a status update to the Data Server and the status GUI.

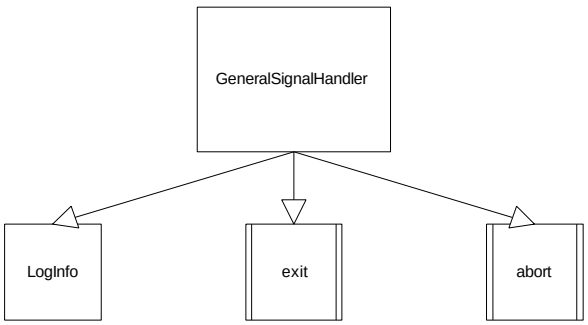

Figure 65. GeneralSignalHandler

Function	Description
abort	C library function.
exit	C library function.
LogInfo	Logs informational messages to the status log and/or standard output.

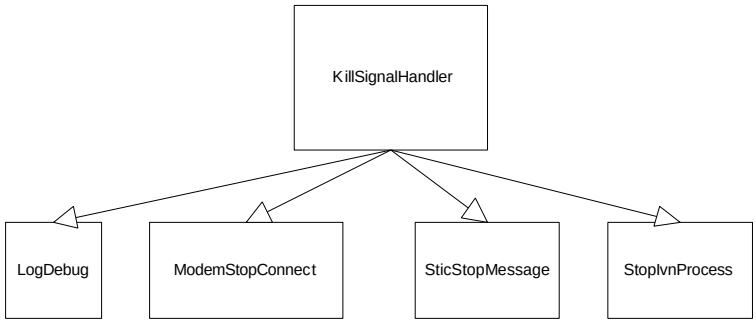

Figure 66. KillSignalHandler

Function	Description
LogDebug	Logs debug messages to the status log and/or standard output.
ModemStopConnect	Interrupts a modem dial connection if in progress.
SticStopMessage	Stops a STIC message transmission if in progress.
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.

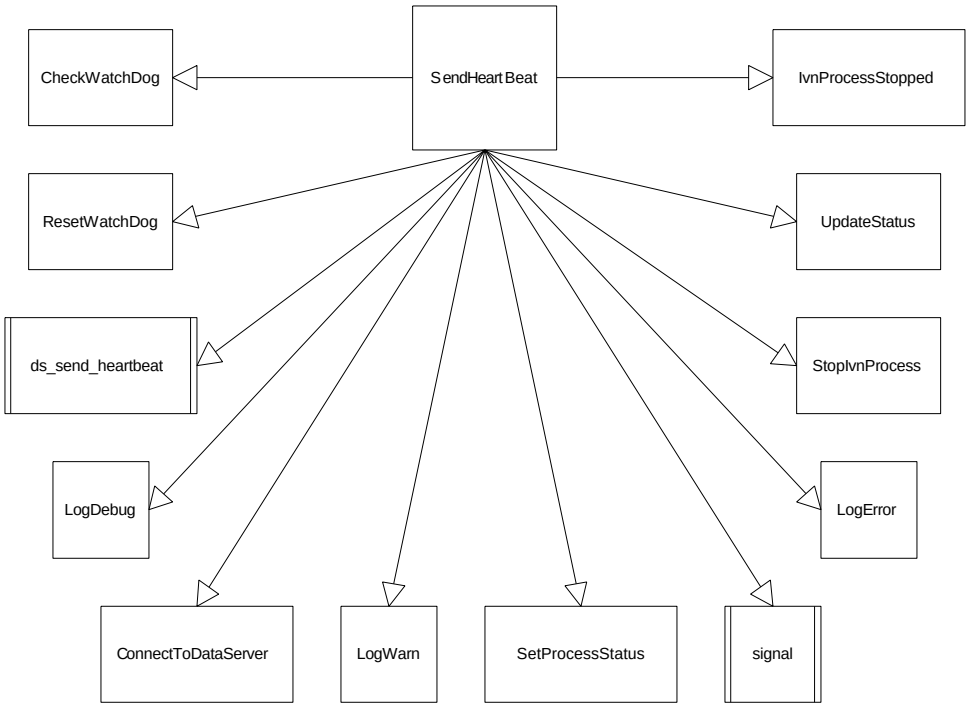

Figure 67. Send Heartbeat

Function	Description
CheckWatchDog	Checks for expiration of the IVN watchdog timer.
ConnectToDataServer	Establishes the socket connection to the Data Server process.
ds_send_heartbeat	Data Server library function. Sends the process status to the Data Server.
IvnProcessStopped	Returns true if the IVN process has been stopped.
LogDebug	Logs debug messages to the status log and/or standard output.
LogError	Logs error messages to the status log and/or standard output.
LogWarn	Logs warning messages to the status log and/or standard output.
ResetWatchDog	Resets the IVN watchdog timer.
SetProcessStatus	Sets the IVN process status to the specified state (OKAY, WARNING, or ERROR).
signal	C library function.
StopIvnProcess	Sets a flag to indicate that the IVN process has been stopped.
UpdateStatus	Updates the status GUI shared memory.

4.2.4 Data Structures

A number of data structures are defined within the IVN MCS. The most significant are those associated with the IVN communication protocol and the STIC message encoder communication protocol.

4.2.4.1 IVN Communication Protocol Data Structures

The IVN communications protocol defines the format of the real time traffic data that is broadcast to the STIC receivers and IVN units. This protocol defines a highly compressed data format in order to minimize the volume of data transmitted. The protocol has three primary message layers: the STIC Transmission Messages (STMs), the Traffic Information Messages (TIMs), and the

Location References (LRs). Data structures defined within the IVN MCS correspond almost directly to the message components of the IVN communications protocol. The major components of these data structures are described below, from the highest to lowest level message layer. For a detailed description of each field within the structures, refer to the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol*.

The highest level data structures are the STMs shown below. STM data structures are defined for both the link speed data and the incident data. Each STM contains a header and a defined number of Traffic Information Messages (TIMs). For the IVN implementation, only a single TIM is contained within each STM. Arrays (frames) of link speed and incident STMs comprise the formatted real time data which is transmitted to the STIC message encoder. The complete set of STMs which contain all link speeds and all incidents is referred to as the STIC superframe.

```
typedef struct {        /* Link STM */
  STMHEADER stmheader;
  LINKTIM linktim[LNKTIMSPERSTM];
} LINKSTM;

typedef struct {        /* Incident STM */
  STMHEADER stmheader;
  INCIDENTTIM inctim[INCTIMSPERSTM];
} INCIDENTSTM;
```

The format of the STM header is shown below. The start flag is a unique byte sequence used to flag the start of the STM data stream. The sequence number uniquely identifies the STM. Once an STM is created, the STM sequence number changes only when the data within the STM is modified, or at midnight of each day when all STM sequence numbers are reset. This allows receivers to ignore STMs which have been previously processed on a given day.

```
typedef struct {        /* STM header */
  STMFLAG startflag;
  USHORT bytecount;
  USIGN daycode:3;
  USIGN sequence:13;
} STMHEADER;
```

TIMs are defined for both link speed and incident data. Each TIM contains a header and either the speed data for a defined number of links or the incident data for defined number of incidents.

```
typedef struct {        /* Link TIM */
  LINKTIMHEADER timheader;
  LINKTIMDATA data[LNKSPERTIM];
} LINKTIM;

typedef struct {        /* Incident TIM */
  INCIDENTTIMHEADER timheader;
  INCIDENTTIMDATA data[INCSPERTIM];
} INCIDENTTIM;
```

TIMHEADER structures are defined for both link speed and incident TIMS. The TIM header contains information that is common to the links or incidents contained within the TIM. Although the structure of the link speed and incident TIM headers differ somewhat, they both include a location reference header for the TIM data and the number of data items contained in the TIM.

```
typedef struct {        /* Link TIM header */
  UCHAR msgcode;
  USIGN :5;
  USIGN msgtime:11;
  USIGN fmtcode:3;
  USIGN numsegments:5;
  LRHEADER lrheader;
  UCHAR numlinks;
} LINKTIMHEADER;

typedef struct {        /* Incident TIM header */
  UCHAR msgcode;
  USHORT msgtime;
  LRHEADER lrheader;
  UCHAR numincidents;
} INCIDENTTIMHEADER;
```

The LRHEADER defines the location of the origin used for relative location referencing of links and incidents. The locations of the links and incidents within a TIM are expressed as offsets from this origin unless the distance from the origin requires an absolute location reference. The *lrseqtype* field defines the location reference type of the TIM. The origin data is a bit field which contains the longitude and latitude of the origin in compressed format.

```
typedef struct {        /* Location reference header */
  USIGN lrmscode:4;
  USIGN lrtype:4;
  USIGN lrseqtype:1;
  USIGN lrseqid:2;
  USIGN datum:3;
  USIGN origintype:2;
  ORIGINDATA origindata;
} LRHEADER;
```

Each TIMDATA structure contains the link speed or incident data for a single link or incident. For link speed TIMs this includes the link location, the link speed, and the TransGuide link ID. For incident TIMs this includes the incident type, the incident location, start time, end time (not currently used), the TransGuide link ID, and a unique incident ID code. Note that the TransGuide link ID fields and incident ID field in these data structures are used by the IVN process but are not part of the IVN protocol and are not included in the data transmitted to the STIC message encoder.

```
typedef struct {        /* Link TIM data */
  LOCATION location;
  UCHAR speed;
  TGLinkID linkid;
} LINKTIMDATA;

typedef struct {        /* Incident TIM data */
  UCHAR incidenttype;
  LOCATION location;
  USHORT begintime;
  USHORT endtime;
  TGLinkID linkid;
  int incidentid;
} INCIDENTTIMDATA;
```

The LOCATION data structure is a union of data structures defining each of the four possible location reference types: local point, local link, global point, and global link. Only one of these types is valid for a particular location and is indicated by the LR header *lrtype* field.

```
typedef union {         /* Location information */
  LOCALPOINT localpoint;
  GLOBALPOINT globalpoint;
  LOCALLINK locallink;
  GLOBALLINK globallink;
} LOCATION;
```

Each of the four location data structure types is shown below. The LOCAL location data structures define longitudes and latitudes that are locally referenced from the origin stored in the location reference header. The GLOBAL location data structures use global referencing for longitudes and latitudes, which are recorded as absolute locations. The POINT location data structures define the location of a point and include a single latitude, longitude and street level code. Points are used only to describe the locations of incidents. The LINK location data structures define the location of links and include both a starting and ending latitude, longitude and street level code. Links are used to describe the locations of links or incidents. Each type of location data structure contains fields defining street name information which may be optionally included in the location reference.

```
typedef struct {      /* Locally referenced point */
  USIGN lrmscode:4;
  USIGN lrtype:4;
  short longitude;
  short latitude;
  char level:4;
  USIGN :2;
  USIGN streetflag:1;
  USIGN streetdata:1;
  STREET street;
} LOCALPOINT;

typedef struct {      /* Locally referenced link */
  USIGN lrmscode:4;
  USIGN lrtype:4;
  short startlon;
  short startlat;
  short endlon;
  short endlat;
  char startlevel:4;
  char endlevel:4;
  USIGN :6;
  USIGN streetflag:1;
  USIGN streetdata:1;
  STREET street;
} LOCALLINK;

typedef struct {      /* Globally referenced point */
  USIGN lrmscode:4;
  USIGN lrtype:4;
  long longitude;
  long latitude;
  char level:4;
  USIGN lrseqid:2;
  USIGN streetflag:1;
  USIGN streetdata:1;
  STREET street;
} GLOBALPOINT;

typedef struct {      /* Globally referenced link */
  USIGN lrmscode:4;
  USIGN lrtype:4;
  long startlon;
```

```
  long startlat;
  long endlon;
  long endlat;
  char startlevel:4;
  char endlevel:4;
  USIGN :4;
  USIGN lrseqid:2;
  USIGN streetflag:1;
  USIGN streetdata:1;
  STREET street;
} GLOBALLINK;
```

4.2.4.2 STIC Message Encoder Communication Protocol Data Structures

The IVN Process communicates to the STIC message encoder in accordance with the STIC message encoder communication protocol. This protocol has the external controller (in this case the IVN process) as the master which initiates all communications to the STIC (the slave). All messages received by the STIC result in a response from STIC indicating the success or failure of the STIC to interpret and respond to the message.

The STIC communication protocol defines two primary message types: forward messages issued by the external controller and response messages from the STIC. The IVN uses only a subset of the forward message types available within the STIC protocol: the ADD PACKET, CLEAR PACKETS, and QUERY MARKER messages. The ADD PACKET message is used to add data to the STIC broadcast packet. The CLEAR PACKETS message is used to clear all data from the broadcast packet. The QUERY MARKER message is used to determine when a broadcast packet becomes active (is being transmitted). The data structures defined within the IVN MCS for the STIC message types used by the IVN process are described below. For a complete description of the STIC communication protocol and message types, refer to the *Subcarrier Traffic Information Channel (STIC) Interface Control Document.*

The STICMSG structure defines the format of forward messages from the IVN process to the STIC. Forward messages are delimited by byte arrays which are unique byte sequences which flag the start and end of the STIC message. The message also contains header information, the message data, and a crc which is computed on the start flag, header, and data fields. Note that the data field is optional depending on the forward message type. For the IVN application, the data field is used only for the ADD PACKET message.

```
typedef struct {        /* STIC message structure */
  STICFLAG somflag;
  STICMSGHEAD head;
  STICMSGDATA data;
  USHORT crc;
  STICFLAG eomflag;
} STICMSG;
```

The STICRESP structure defines the format of response messages from the STIC. Response messages are delimited by byte arrays which are unique byte sequences which flag the start and end of the STIC message. The message also contains header information, the message data, and a crc which is computed on the start flag, header, and data fields. Note that the data field is optional. For the IVN application, the data field is included only in responses to the ADD PACKET message.

```
typedef struct {        /* STIC response structure */
  STICFLAG somflag;
```

```
  STICMSGHEAD head;
  STICRESPDATA data;
  USHORT crc;
  STICFLAG eomflag;
} STICRESP;
```

The STICMSGHEAD structure defines the unit ID of the STIC to receive the message (always 0 for the IVN application), the length of the message in bytes, and the message type code.

```
typedef struct {        /* STIC message header structure */
  UCHAR unitid;
  USHORT bytecount;
  UCHAR msgcode;
} STICMSGHEAD;
```

The STICMSGDATA structure defines the format of the data included in the ADD PACKET forward message. It includes a unique sequence number which identifies the packet and the actual message data which for the IVN application is the formatted link speed or incident STMs.

```
typedef struct {        /* STIC message data structure */
  ULONG msgseq;
  STICDATA msgdata;
} STICMSGDATA;
```

The STICRESPDATA structure defines the format of the data in the ADD PACKET response message. It includes the sequence number of the packet that was added and the number of bytes available in the broadcast message for additional packets.

```
typedef union {         /* STIC response data structure */
  ULONG seqnum;
  USHORT bytesleft;
} STICRESPDATA;
```

5. Requirements Traceability

The following table correlates the system requirements, design elements, and acceptance tests. The table demonstrates requirements are met and verified.

NUMBER	REQUIREMENT	SOURCE	DESIGN ELEMENT ALLOCATED TO
IVN-1	The system shall communicate the following types of real-time traffic information to moving vehicles: • link speed data • incident information	RFO-33.3.3	MCS/STIC system
IVN-2	The real-time traffic information shall be derived from the Data Server.	P-2.6.2.3.4	MCS
IVN-2.1	Software running on an IVN master computer shall be used to extract real-time information from the Data Server.	P-2.6.2.3.4	MCS
IVN-2.1.1	The IVN MCS shall transmit IVN process status information to the Data Server every 60 seconds.	derived	MCS
IVN-2.1.2	The IVN MCS shall extract link ID information from the Data Server Realtime Subsystem.	derived	MCS
IVN-2.1.3	The IVN MCS shall extract link location information from the Data Server Realtime Subsystem.	derived	MCS
IVN-2.1.4	The IVN MCS shall extract link speed information from the Data Server Realtime Subsystem.	derived	MCS
IVN-2.1.5	The IVN MCS shall extract incident information from the Data Server Realtime Subsystem.	derived	MCS
IVN-2.1.6	The IVN MCS shall transmit link speed information to the STIC message encoder every 60 seconds.	derived	MCS
IVN-2.1.7	The IVN MCS shall transmit incident information to the STIC message encoder every 60 seconds.	derived	MCS
IVN-2.1.8	The IVN MCS shall transmit locally referenced link speed information to the STIC message encoder in accordance with the messaging protocol defined in the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol*	derived	MCS
IVN-2.1.9	The IVN MCS shall transmit globally referenced link speed information to the STIC message encoder in accordance with the messaging protocol defined in the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol*	derived	MCS
IVN-2.1.10	The IVN MCS shall transmit locally referenced incident information to the STIC message encoder in accordance with the messaging protocol defined in the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol*	derived	MCS

NUMBER	REQUIREMENT	SOURCE	DESIGN ELEMENT ALLOCATED TO
IVN-2.1.11	The IVN MCS shall transmit globally referenced incident information to the STIC message encoder in accordance with the messaging protocol defined in the *TransGuide In-Vehicle Navigation System High Speed FM Subcarrier Communications Protocol*	derived	MCS
IVN-2.1.12	The IVN MCS shall generate a table of TransGuide link information that includes link ID, starting coordinate, ending coordinate, and street name.	derived	MCS
IVN-2.1.13	The IVN MCS shall provide a display of the process status which shall be updated every 60 seconds.	derived	MCS
IVN-2.1.14	The IVN MCS shall be capable of being started and stopped through the IVN process status GUI.	derived	MCS
IVN-2.1.15	The IVN MCS shall provide a display of the status of communication to peripheral systems with which it exchanges data.	derived	MCS
IVN-2.1.16	The IVN MCS shall log informational messages, warning messages, and error messages to a status log file.	derived	MCS
IVN-2.2	The IVN master computer shall be a Sun Microsystems UltraSPARCStation with the following components: • 167 MHz SPARC (RISC) CPU • 128 MB RAM • GB hard disk space • Floppy disk drive • Sun CD-ROM drive • Turbo GX+ graphics • 20" color monitor • 8 port modem server (SCSI attached) • Dual ethernet interfaces • Dual SCSI channels	P-2.3.2.4.1	IVN master computer
IVN-2.3	The master computer shall be located in the TransGuide Operations Center.	P-2.6.2.3.4	IVN master computer
IVN-3	The system shall communicate the real-time traffic information using the Subcarrier Traffic Information Channel (STIC) system which consists of a message encoder/FM subcarrier generator and FM subcarrier receivers.	P-2.6.1	STIC encoder/STIC receiver
IVN-4	The system shall utilize commercially available IVN units to display the real-time traffic information to travelers.	RFO-33.4.1	Alpine/Zexel navigation units
IVN-4.1	The IVN unit shall be composed of the following components: • microprocessor • reconfigurable LCD color display • removable media for data storage • gyroscopic sensor • GPS receiver	P-2.6.2.3.5	Alpine/Zexel navigation units

NUMBER	REQUIREMENT	SOURCE	DESIGN ELEMENT ALLOCATED TO
IVN-4.2	The IVN unit shall accept real-time traffic data input from the STIC receiver.	P-2.6.2.3.5	Alpine/Delphi navigation units
IVN-4.3	The IVN unit shall provide a means for the traveler to enter a destination of the following types: • address • intersection • place or point of interest • previous destination	P-2.6.2.3.5	Alpine/Delphi navigation units
IVN-4.4	The IVN unit shall calculate the shortest time route, based on the real-time information from the STIC receiver, from the current location of the vehicle to the traveler-entered destination.	RFO-33.4.2	Alpine/Delphi navigation units
IVN-4.5	The IVN unit shall communicate the route information using a map display, guide display, and audible prompting.	P-2.6.2.3.5	Alpine/Delphi navigation units
IVN-4.6	The IVN unit guide display shall present the following information: • distance to an upcoming turn, • direction of an upcoming turn.	P-2.6.2.3.5	Alpine/Delphi navigation units
IVN-4.7	The IVN unit shall provide the following audible prompts: • warning of an upcoming turn, • indication to make a turn.	RFO-33.4.2	Alpine/Delphi navigation units
IVN-4.8	The IVN unit map display shall show the current location of the vehicle and the calculated route on an annotated map of the surroundings.	P-2.6.2.3.5	Alpine/Delphi navigation units
IVN-4.9	The information used to generate the map display shall be derived from the San Antonio, Texas Metropolitan Area database supplied by Navigation Technologies.	P-2.6.2.3.5	Alpine/Delphi navigation units
IVN-5	The system shall utilize the IVN units of at least two manufacturers.	RFO-33.4.1	Alpine/Delphi navigation units

APPENDIX A - IVN SYSTEM ENGINEERING DRAWINGS