Part II

Ground Terminal Expert (GTEX)

Expert System Diagnostics for a 30/20 Gigahertz Satellite Transponder

A final report summarizing research
on
Contract NAG3-923

completed at

The Univerisity of Akron
Electrical Engineering Department
Akron, Ohio 44325

for

NASA Lewis Research Center
21000 BrookPark Road
Cleveland, Ohio 44135

submitted by

Dr. John Durkin Associate Professor of Electrical Engineering
Richard Schlegelmilch Masters Student in Electrical Engineering
Donald Tallo Masters Student in Electrical Engineering

March 31, 1992

ABSTRACT

A research effort was undertaken to investigate how expert system technology could be applied to a satellite communications system. The focus of the expert system is the satellite earth station. A proof of concept expert system called the Ground Terminal Expert (GTEX) was developed at The University of Akron in collaboration with NASA Lewis Research Center.

With the increasing demand for satellite earth stations, maintenance is becoming a vital issue. Vendors of such systems will be looking for cost effective means of maintaining such systems. The objective of GTEX is to aid in diagnosing of faults occurring with the digital earth station.

GTEX was developed on a personal computer using the Automated Reasoning Tool for Information Management (ART-IM) developed by Inference Corporation. Developed for the Phase II digital earth station, GTEX is a part of the Systems Integration Test and Evaluation (SITE) facility located at NASA Lewis Research Center.

TABLE OF CONTENTS

CHAPTER I

STATEMENT OF PROBLEM

1.1 Background

Extensive development of satellite communications is currently under way at the NASA Lewis Research Center. Using proof-of-concept subsystems and components, (IF switch matrices, solid-state amplifiers, traveling-wave tube, high-power amplifiers and low-noise receivers) a Ka-band satellite communication network simulation known as the Satellite Integration, Test and Evaluation (SITE) facility has been developed. This facility allows modulated data to be used to characterize the effect of microwave components through a Time Division Multiple Access (TDMA) burst terminal and an earth station network.

In 1989, the NASA Lewis Research Center initiated a grant with the University of Akron to investigate the feasibility of applying expert system technology to satellite communications. The SITE test bed provides the satellite communication benchmarks. The Ground Terminal Expert (GTEX) system was developed as a demonstrational prototype. GTEX addresses faults associated with the digital earth station.

1.2 Characteristics of the Digital Earth Station

The digital earth station is a major and one of the more complex elements of the satellite communications network. The earth station is responsible for acquiring satellite and network timing, maintaining synchronization of the network, and transmitting and receiving data from other earth stations in the network. Each SITE earth station, shown in Figure 1.1, consists of a system clock, timing and control circuits for both transmitting and receiving, first-in, first-out memories(FIFO), individual user clocks and associated control circuits, command processor microcomputer, a user interface controller, and a serial minimum-shift(SMSK) burst modulator and demodulator (Ivanic, et al. 1989).

In the SITE earth station, users are simulated by a bit-error-rate test set consisting of a data generator (transmitting user) and a data checker (receiving user). A controlling computer creates realistic traffic patterns with users of varying data rates entering and leaving the system. A bit-error-rate (BER) figure, a performance measure of the overall satellite system, is used to determine the degradation of data. By knowing the degree of degradation the end-user can decide to either tolerate it or make necessary compensations (Shalkhauser 1988).

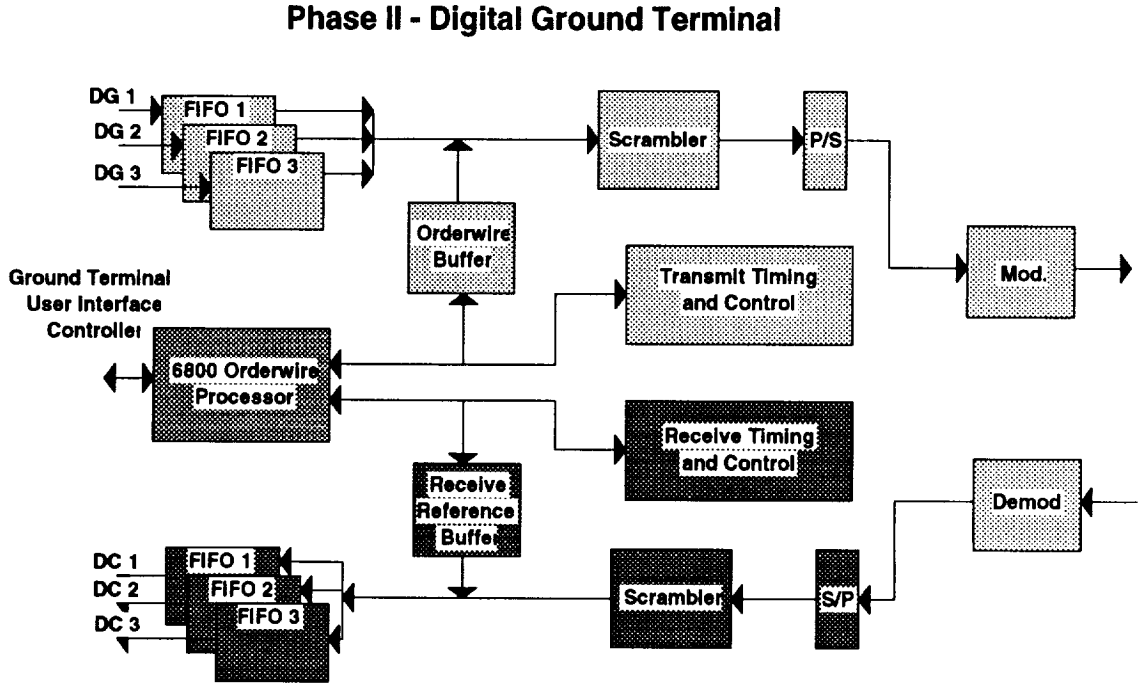

Figure 1.1 -- SITE digital earth station

1.3 Issues Regarding the Digital Earth Station

The SITE earth station is limited to the NASA Lewis user community. One earth station having similar network characteristics as those of the SITE station, and a vast user community, is the very small aperture terminal (VSAT). "VSAT represents technology innovation that provides reliable transmission of digital information for data, voice and even video by satellite using comparatively small antennas." (Levenburg 1991)

The most difficult problem faced by the designers of the personal VSAT is the realization of acceptable performance levels while achieving a high level of subsystem integration at low cost (Tsang et al. 1991). One expert estimates the cost to operate "a hypothetical 100-node VSAT network per month at roughly

$22,050, which include $6,500 for maintenance of such a system." (Case 1990) Case further states, "For vendors to be successful, they will have to provide the application software, network management and maintenance service and hardware." He continues by saying, "The hardware part of the package will become almost a commodity." Therefore, "The turnkey service and the applications software will become the key factors used by companies to select a vendor." Vendors will be looking for methods to maintain these stations in the most cost effective means possible.

1.4 Expert System Solution

One method of reducing maintenance cost is by minimizing the time required by technicians to diagnose and maintain the earth station. This time could be minimized by having an expert system focus the diagnostic steps, eliminating unnecessary procedures.

GTEX reduces the time spent diagnosing the faults. A technician is not using costly time searching reference manuals for specifications and procedures contain within the knowledge base of GTEX. GTEX guides the procedures necessary to diagnose the fault. Since GTEX contains information about all faults, the probability of not diagnosing the correct fault is minimal. Less time is spent diagnosing errors, which results in less down-time of the communication link. These time saving procedures results in a cost savings for both the consumer and service companies.

The main concern with any digital earth station is the errors associated with the transmitting and receiving of user data. Since the SITE earth station displays this characteristic, the GTEX prototype focused on diagnosing such errors.

1.5 The Development Environment

GTEX was developed using the Automated Reasoning Tool for Information Management(ART-IM) by Inference Corporation. ART-IM is a C-based toolkit for the development of rule-based, or frame-based, expert systems (ART-IM 1991). GTEX was developed on a personal computer(PC) running under MS-DOS.

ART-IM supports three programming styles; procedural, rule-based and object-oriented. The procedural language supported by ART-IM provides basic function calls, and allows simple interactions and conditionals to be performed. The rule-based structure uses the rule as the fundamental unit. Reacting to changes in the working memory, the rule can then fire or execute based on the

dynamic order of the changes that occur. Objects in ART-IM are represented by a schema. Control of an object is managed by sending a message to that object. An object reacts to a message by searching itself for an appropriate method and executing the actions associated with that method.

The ART-IM procedural language can be extended using the 'C' language. User functions can be written in 'C' and included with the ART-IM program that can be used like any other ART-IM function. Since functions defined in 'C' are complied versus interpreted, the result is faster execution. This capability of ART-IM was important in the development of the user-interface discussed in this report.

The hardware requirements of GTEX include:

- IBM/AT, 386, 486 or compatible computer
- Minimum of 2 Megabytes of RAM
- Hard-disk with at least 1 Megabyte available
- VGA color monitor
- Microsoft compatible mouse

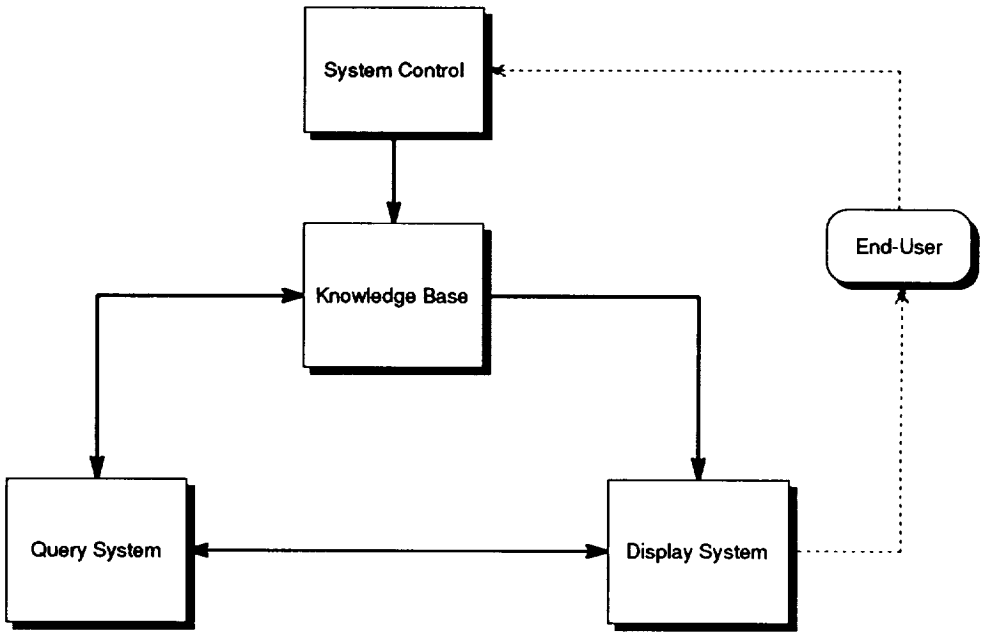

Figure 1.2 -- GTEX architecture

1.6 Overview of GTEX

The GTEX system, shown in Figure 1.2 is responsible for the following:

- Fault Verification
- Fault Isolation
- Fault Recovery Recommendation

The diagnostic procedure begins by determining the initial configuration of the system. Information about each user channel regarding transmit and receive data channels, bit error rate (BER) and data rate is gathered. General status information about the SITE ground terminal is also obtained. Each user data channel BER measurement is verified to determine if the tolerance limit has been exceeded. If tolerance limits are exceeded, the end-user is informed of the discrepancy and isolation of the fault begins.

Three levels of isolation are performed. The initial stage of isolation determines which side, transmit or receive, of the user channel is causing the error. Isolation then continues to determine which of the corresponding subsystems is in error. Finally the corresponding circuit board is isolated. Once the fault has been isolated and end-user informed, GTEX can be requested to recommend an appropriate action.

CHAPTER II

GTEX ARCHITECTURE

2.1 Introduction

The GTEX architecture, shown in Figure 1.2, is comprised of four major systems; Knowledge base, Query, Control, and Display systems. The role of each system is discussed.

2.2 Knowledge Base

The essential difference between an expert system and a conventional program is that the expert system processes' knowledge while the conventional program processes' data. One of the primary components of any expert system is the knowledge base. The knowledge base is the collection of expert knowledge about a given domain. The GTEX knowledge base is a collection of rules and objects associated with diagnosing the digital earth station.

Rules represent the operational knowledge elements. They describe actions that should occur based on conditions of the working memory. Rules also express heuristics of the given domain. The rule construction consists of left and right - hand side elements. A typical rule from GTEX, is shown in its natural langauge form in Figure 2.1.

RULE	Ground Terminal Isolated
IF	The Transponder is not Connected
AND	The Attenutators are not Connected
THEN	The Ground Terminal is Isolated

Figure 2.1 - GTEX rule in natural language format

Objects allow a structure to be defined for a complex domain. Objects provide support for data abstraction, knowledge encapsulation, reusability, and extensibility. An overview of object-oriented programming will be presented in Chapter 5.

Individually, each representation technique has many advantages over traditional programming techniques. Used together their advantages are further enhanced. Augmenting rules with objects allows further refinement of knowledge than with rules alone. Objects allow the knowledge to be structured, exploiting data abstraction and encapsulation principles in the application. Using objects with rules provides a more powerful paradigm for reasoning about objects. (Stipp 1990)

2.3 Query system

The query system is responsible for obtaining data upon request from the knowledge base. Two forms of data acquisition are possible: simulation and user dialog. Since the digital ground terminal is under direct computer control, by allowing GTEX to retrieve data from a simulator the dynamic retrieval of data is portrayed. User dialog provides a mechanism for retrieving data from the end-user. The end-user is prompted for data whenever interaction is requested by the end-user or the simulators' inability to yield desired data. GTEX requires all pieces of data. The current implementation includes no provisions for reasoning about unknown pieces of data.

2.4 Control system

End-users' require some convention for controlling computer program execution. Though computers are becoming more common place, people are still intimidated by them. By not providing mechanisms for control, end-users' feel constrained, increasing their intimidation of computers. Feeling intimidated, the end-user will forgo using your program and regress to alternative methods of performing the same task. Confidence in a computer program is increased if the end-users' feel they are controlling execution. Accordingly, the developer has to be careful not to give too much control to the end-user. If too much control is given, the end user is susceptible to making mistakes and causing system errors.

GTEX gives the end-user control over the system by a series of push-buttons and display controls. Push-buttons provide complete control of GTEX execution. Display controls allow the end-user to examine the digital ground terminal at various levels of detail. Control possibilities at given instances are limited by the program execution. Permitting the program to control available actions prohibits the end-user from making serious errors.

2.5 Display system

A graphical interface permits the end-user to interact with the GTEX system. This type of interface is also known as a direct manipulation interface (Schneiderman 1987). GTEX incorporates a form of the direct manipulation interface called a point-and-do interface. Point-and-do direct manipulation interfaces give the end-user the capability of pointing to commands and objects on the display to provide the necessary actions. There are some key benefits associated with this type of interface. Associated with text-based systems, end-users could introduce typing and memory errors. These errors are caused by the end-user typing in the necessary commands to perform essential actions. Given complex and rigid command syntax, end-users can make errors which might be considered to him as small. However, these errors stop computers from carrying out commands (Krull & Rubens 1986). Another helpful aspect of point-and-do interaction is its reducing end-user memory load. By not having to remember commands and file names; the end-user can capitalize on their spatial skills rather than requiring them to work through more difficult abstract verbal reasoning processes(Heckel 1984; Hemenway 1982; Shamonsky 1985; Muter & Mayson 1986).

CHAPTER III

SYSTEM OPERATION

3.1 Introduction

The GTEX knowledge base is responsible for the following:

- Fault verification
- Fault isolation
- Fault recovery recommendation

The knowledge base also performs actions, such as displaying key-information using intermediate dialog and dynamically updating screens based on conditions within working memory. A detailed discussion of the diagnostic procedures is presented.

3.2 Assumptions

The following assumptions are made during GTEX execution:

The GTEX system is used as a diagnostic tool, aiding a technician in (1) verifying that a fault has occurred and (2) performing the necessary tasks to remedy the problem.

The technician is knowledgeable about performing required tasks, such as isolating the earth station from outside sources, replacing necessary components, and knowing the necessary procedures for running the earth station. It is also assumed that the technician is capable of obtaining data from various sources, which include the earth station microcomputer system monitors.

The transmit and receive users are located in the same earth station chassis.

If the earth station is connected to the SITE transponder, it is assumed that the transponder is configured with the traveling wave tube amplifier operating in low mode. Other modes will change how faults are verified.

3.3 Fault verification

Fault verification determines if the earth station is exhibiting a fault under the current conditions. By verifying a fault, required working memory elements needed for isolation are initialized.

Initially, GTEX determines the current operating configuration of the earth station, which includes, the usage of transponder and noise attenuation (if any), user channel operating configuration and system status. The transponder is another component of the SITE testbed. The transponder simulates the operation of the satellite in orbit. Noise can be inserted in the signal by an external source. The noise unit is always used in conjunction with the transponder; but it can also be used by itself. The operating configuration of each channel includes: (1)transmit channel, (2)data-rate, (3)receive channel, and (4)the current bit error rate(BER). The acquisition status of the earth station is obtained. Acquisition is the state of the earth station where the transmit and receive signals are in sync with one another. With out acquisition, the data received by the earth station is unusable.

Once the information is obtained verification begins. The first procedure in verifying that a fault exists is determining the current signal to noise ratio (E_b/N_0). The E_b/N_0 is important in determining the performance of the earth station. The value of the E_b/N_0 is obtained from the attenuation settings. A linearized formula is used in predicting the E_b/N_0. Once obtained a bit error rate(BER) measurement can be extrapolated for the given conditions of the transponder. The BER is a measurement of the bits received in error versus the total of number of bits received.

$$BER = \frac{\text{Bits received in error}}{\text{Total number of bits received}}$$

A typical BER for the SITE system is approximately $1x10^{-7}$. If the earth station is running in 'back-to-back' mode for testing purposes the expected BER of the earth station is '0'.

If the BER extrapolated from the current E_b/N_0 is greater than this value, reliable communications can not be achieved. If the expected BER is below the required $1x10^{-7}$ verification of each user channel is completed. The expected BER is compared to the BER measured by each channel. If the BER for a given channel is higher than the expected, it is assumed to be in error. With this

implementation of GTEX, fault isolation will occur only if a single user channel is in error.

3.4 Fault isolation

Fault isolation provides the procedures for locating the source of the fault. The isolation procedures selectively narrows the source of the problem to board-level. These procedures focus the search, thus reduces the time required by the technician to diagnose the fault.

The first step in isolating a fault is to verify the operation of the earth station. This is completed by isolating it from all external signals, i.e. transponder. To verify the operation, the technician is requested to run the earth station using the initial channel configuration parameters. The earth station is assumed to be the problem source if the fault reoccurs, or the initial configuration indicates that the earth station is isolated.

With the initial configuration including an external signal and the isolated earth station is functioning properly, the technician is prompted by GTEX to again verify the initial configuration. If the fault reappears the source of the problem is assumed to be the external signal.

Assuming the earth station as the problem source, the next procedure performed is locating the fault on either the receive or transmit side of the earth station. The fault is located by performing a cross test, with the transmit and receive signals with one of the satisfactorily operating channels. Since the diagnostic procedures for the transmit and receive sides of the earth station are similar, the GTEX prototype demonstrates isolating faults that are located on the transmit side.

Finding the fault on the transmit side GTEX continues. The transmitting user of the system is tested to see if they are the source of the error. The same type of cross pattern is applied to the transmitting user. The earth station is designed in such a manner that the users of the system can also run in a back-to-back mode. If the fault still occurs, the transmitting user system is assumed to be in error.

The other components in the signal path are checked if the fault does not occur with the transmitting user. The serial-to-parallel board is the next suspect in the isolation phase. To verify the operation of the serial-to-parallel board, it is replaced by a serial-to-parallel board taken from another data channel. The system operation is once again verified. If the system is verified the serial-to-parallel board is determined to be the source of the error. Otherwise, then the transmit first-in first-out(FIFO) board is assumed to be the source.

To verify the operation of the FIFO, a transmit test FIFO board is required. The test board acts like a receiving user-checking the transmitted signal. The system is verified. GTEX ends isolation at this point, if errors still occur. These types of errors are outside the scope of GTEX. GTEX then informs the technician about this conclusion. If no errors occur, the FIFO board is assumed to be at fault.

3.5 Fault recovery recommendation

Recovery recommendation provides the ability to either, (1) display the necessary procedures for correcting the fault, or (2) display known system status about the error that can not be determined. Providing the necessary procedures, the technician can perform the necessary steps for repairing the earth station, with little effort.

If GTEX can not locate the error, the technician is responsible for completing the diagnoses. By giving the technician information about the error, procedures will not be duplicated, thus saving maintenance time.

The recommendation is based on the type of error. If GTEX is able to diagnose the error, the recommend recovery procedures are presented to the technician. If a fault cannot be diagnosed, key-elements of working memory are displayed. These elements include: user channel configurations and the various levels of isolation completed by GTEX.

CHAPTER IV

IMPLEMENTATION

4.1 Introduction

The design specifications of GTEX required that each system function independently. A message structure was implemented to establish the communication between the individual systems. The methods used in implementing each system and message structure is described.

4.2 Message system

A standard communications protocol was defined to provide the needed interaction between the GTEX subsystems. By developing a standard communications scheme, the necessity of each subsystem being observant of other subsystems' formalisms is minimized. The calling subsystem only needs to tell the communications manager which subsystem is the receiver of the message. This standard allows subsystem modification without affecting other written code.

The constructed scheme is modest. A typical call consists of a receiver and a corresponding message. A typical call to the communications manager follows.

```
Send-Message Receiver "Message"
```

Figure 4.1 shows the possible communications paths defined by the communications manager. The communications manager is written in the 'C' programming language and can be called from either the ART-IM or C environments. The calling procedures follow.

ART-IM syntax

```
(send-message receiver "message")
```

C syntax

```
aofnSendMessage(receiver, "message")
```

The 'receiver' is a predefined constant symbolizing an individual subsystem. Table 4.1 itemizes the available constants. The 'message' is unique to the individual receivers. Each message structures will be discussed in the appropriate subsystems descriptions later in this report.

Symbol	Subsystem
Display-System	Display
Query-System	Query
Control-System	Control

Table 4.1 -- Receiver Constants

The communications manager associates the receiver constant with the corresponding function. A call to the receiver function is then generated, including the 'message' as a parameter. The call to the receiver is then executed. The receiver is then responsible for parsing the message and acting accordingly.

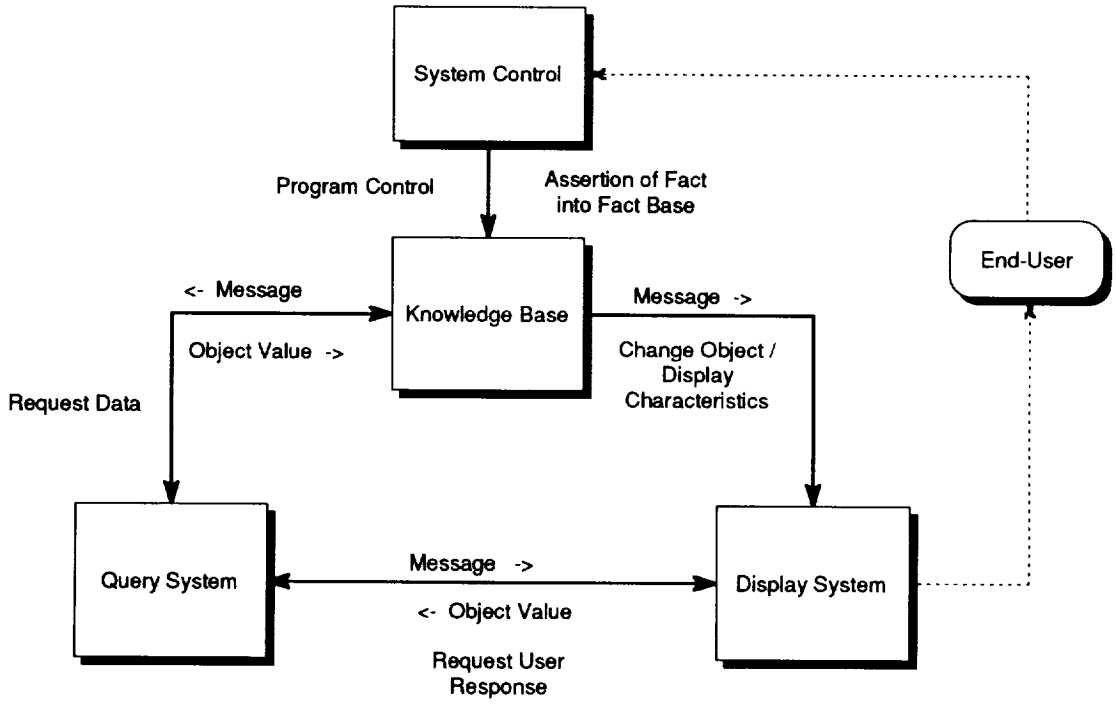

Figure 4.1 -- GTEX Message Structure

4.3 Knowledge base

Several rule classifications were used in the design of the GTEX rules: initialization, verification, isolation, recommendation and miscellaneous rules. Each classification is described in Table 4.2. Each of these classifications provide the framework for the system operation. The framework is composed of fault verification, isolation and recovery recommendation stages. A summary of their operation is presented; a detailed discussion is presented in chapter3.

Rule Classification	Description
Initialization	Provides GTEX with the configuration parameters for the current scenario
Verification	Verifies that a fault exists based on current parameters
Isolation	Performs isolation of fault to board-level in digital ground terminal
Recommendation	Recommends appropriate action based on fault isolated
Miscellaneous	Provides additional support to GTEX (ie. Rules for changing the display)

Table 4.2 -- Rule Classifications

Initialization rules determine the current configuration and status of the earth station. Based on these facts, the condition of the earth station is determined. If the earth station is found to be operating unsatisfactorily, the problem source is isolated. The isolation rules provide the necessary steps for isolating the fault to board level. Once the fault is isolated the recommendation rules are responsible for notifying the technician of the procedures to remedy the problem. The miscellaneous rules are responsible for system initialization, such as the display, and controlling the system.

Each rule classification is responsible for changing conditions of working memory based on previous findings. Each rule classification is responsible for changing the working memory conditions as to the screen to be displayed, updated etc. By giving all this responsibility to each rule the system would of become unmaintainable. Therefore, each rule classification was subdivided into smaller fragments called rule categories, to elevate this problem. The following rule categories subdivided each classification: display, diagnostic, demon. Each

rule category has its own salience or relative priority used in scheduling the rule for firing. Table 4.3 lists the salience values assigned to the individual categories. Each rule category is described.

Rule category	Salience
Display Rules	200
Demon rules	100
Diagnostic rules	0

Table 4.3 -- Rule Category Salience

4.3.1 Display Rules

Writing graphical user interfaces in procedural languages has always been difficult because such interfaces are inherently "interrupt-driven." Interrupt-driven applications are hard to express in higher-level languages because of the detailed nature of the interrupts involved. Most high-level procedural languages wait for the user to respond to an input request. This approach is clearly inappropriate for mouse interaction.

One way of coping with the interrupt-driven nature of graphics applications is to use forward-chaining inference rules to express how the application should react to various events, including mouse interactions. This approach is an ideal means of specifying interrupt-driven interactions because the rules are always ready to fire as soon as the pattern appears in working memory. Hardware interrupts, such as mouse interactions, need only manifest themselves as changes in working memory and the system can readily respond to mouse events. Since a data representation for mouse events must already exist, it is trivial to represent the mouse event at a much higher level of abstraction than "The left mouse button was released at pixel location (150, 250)," which is essentially what the hardware provides.

This technique is quite different from conventional object-oriented graphics programming, in which changes in imagery are invoked by sending object instance messages to move or change color. Similarly, mouse interactions are represented by the system's automatically sending a picked message to an object instance. Both approaches are quite complete, but the inference-based approach is substantially more expressive than the purely object-oriented one. The result is that fewer rules (or statements) are required to express an application (Harris 1990).

```
(defrule Display-Screen
        (Declare (salience ?*display-salience*))
        ?fact <- (display screen ?screen)
        (schema ?active-display
                (Active True))
=>
        (delay 2)
        (modify (schema ?active-display
                (active FALSE)))
        (modify (schema ?screen
                (active TRUE)))
        (Screen (get-schema-value ?screen Display-name))
        (send-message Control-System "SubLevel-Display")
        (retract ?fact))
```

(a)

```
(DEFRULE Ground-Terminal-Is-Isolated
        (Continue)
        (SCHEMA KB-Transponder
                (connected FALSE))
        (Schema KB-Attenuator-1
                (connected FALSE))
        (schema KB-Attenuator-2
                (connected FALSE))
=>
        (MODIFY (SCHEMA KB-Ground-Terminal-Status
                        (Isolated TRUE)))
        (message 104))
```

(b)

```
(DEFRULE Cancel-Input
        (DECLARE (SALIENCE ?*demon-salience*))
        ?fact <- (Continue)
        (SCHEMA ?schema
                (?slot-name CANCEL))
=>
        (retract ?fact)
        (modify-schema-value ?schema ?slot-name UNKNOWN)
        (send-message control-system "set-continue"))
```

(c)

Figure 4.2 -- GTEX rule example
(a) Display Rule (b) Diagnostic Rule (c) Demon Rule

The display rules are responsible for control of all graphic displays. Display rules are activated either by facts asserted by user control or a dynamically changing condition in working memory. The active rule sends an appropriate message to the display system with the current request. Display rules allow modularity in programming. The knowledge base can be altered without

affecting the currently created displays. An example of a display rule is shown in Figure 4.2.a. The display rules have the highest priority thus insuring the user interface is always current.

4.3.2 Diagnostic Rules

It is important to represent and use domain knowledge in a way in which the domain expert represents and uses it. The closer the expert system approximates the expert's mental representation, the easier it will be for the expert to tell you if the expert system you are building is accurate (Martin et al. 1988).

The diagnostic rules are the foundation of the knowledge base. These rules perform the fundamental steps required in diagnosing the digital ground terminal. An example of a diagnostic rule is shown in Figure 4.2.b.

4.3.3 Demon Rules

> Demon: A procedure that is activated for the purpose of accessing or changing values in a data base. It is a type of suspend process that is "waiting" for a certain kind of event to occur, such as a certain kind of update operation on a data base. The demon activates when the special event occurs, performs the job, and either terminates or suspends while awaiting another event. Demons are typically used to make inferences as new information comes into the data base, to perform bookkeeping task of some kind, or to recognize important occurrences.
> Definition from: Facts on File Dictionary of Artificial Intelligence

Sitting dormant in the knowledge base are demon rules. The responsibility of demon rules is to inject vital information into the working memory. An example of a demon rule is shown in Figure 4.2.c. Demon rules were implemented since the object system of ART-IM does not handle dynamic message passing. The objects are unable to send messages based on dynamically changing values. Demon rules provide this capability.

4.3.4 Object system

The ART-IM environment provides use of an object system represented by schemas. The schema hierarchy shown in Figure 4.3, represents the overall object system constructed. The knowledge base hierarchy represents the model of the digital earth station used in the inference process. The dialog hierarchy allows the digital earth station model to be associated with dialog requests. The simulator hierarchy is created dynamically. A detailed description of the object system is presented in chapter 6.

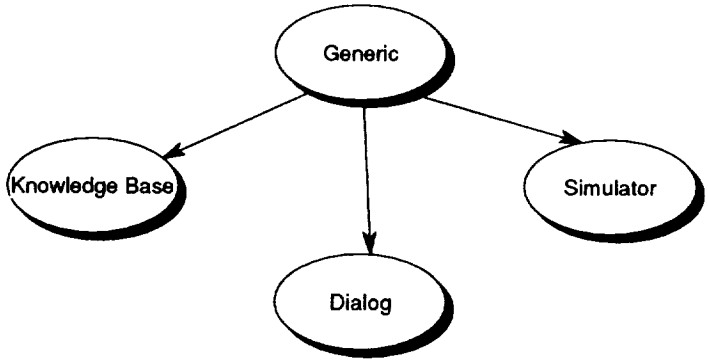

Figure 4.3 -- Overall Schema Hierarchy

4.3.5 Added functionality to the knowledge base

User defined procedures increase the functionality of the knowledge base. These procedures allow for compact rule construction, thus increasing rule understanding. Procedures can be created by extending the ART-IM procedural language by allowing ART-IM commands and specialized functions to be coded and executed like other predefined system functions. A short description of the ART-IM extended procures follows.

Get-Schema-Parent	Allows a child to be able to determine who its parent is and returns it.
String-Name	Takes an ART-IM symbol and returns a quoted string of the symbol.
Message	Sends a message to the display system to display a given message number.
Value	Query a value of the object:slot from the query system.
Screen	Sends message to the display system to display a named screen.
Update-Screen	Sends message to the display system that an update should be performed based on the message sent.
Reset-GTEX	Resets the knowledge base.
Send-message	The handler which directs the message to the appropriate system.

Integer-to-string	Returns a string representation of the integer which was given as an argument.
Mouse-handler	This is the function described for the asynchronous function processing all mouse input.
Initialize-simulator	This function accepts the filename of the data file to be parsed and interpreted as the simulator file.
Save-Environment	This allows GTEX to be reset, by storing the initial environment. Called upon entering GTEX.
Clean-Exit	Performs house keeping, frees memory, etc.

4.3.6 Controlling the knowledge base

The end-user is capable of controlling the execution of the knowledge base. This is accomplished by the existence of the 'continue' fact. This fact is used in rules requiring an unknown piece of knowledge. Canceling the request, the 'continue' fact is retracted. If the end-user elects to continue execution, the 'continue' fact is re-asserted. Intermediate dialog, which provides the end-user with system status messages, is also supplied. This type of dialog also allows the end-user to stop system execution. This dialog is important when the system is responding to simulator input, thus requiring no end-user interaction.

```
(DEFRULE Bit-Error-Rate
    (Continue)
    (schema receive-FIFO-display
                (active TRUE))
    (SCHEMA ?Channel
        (INSTANCE-OF KB-User-Channel)
        (Transmit-Channel ?value &: (integerp ?value))
        (Receive-Channel ?value2 &: (integerp ?value2))
        (BER UNKNOWN)
        (Data-Rate ?rate &: (floatp ?rate)))
=>
    (message (get-schema-value ?channel BER-Rqst-msg))
    (modify (schema ?channel
                    (BER =(value (string-name ?channel) BER)))))
```

Figure 4.4 -- Example of 'Continue' fact

Giving end-users this type of system control provides for exploration of the individual displays and resetting or exiting the knowledge base at their discretion. Figure 4.4 is an example of the 'continue' fact in use. The assertion of

the 'continue' fact is a simple but effective way of controlling the execution of the knowledge base.

4.4 Query system

The query system is a combination of 'C' source code and ART-IM constructs. The ART-IM schema system provides an efficient method of storing and retrieving data. The query system accepts a message in the form of an object-attribute pair, i.e.

"Object Attribute"

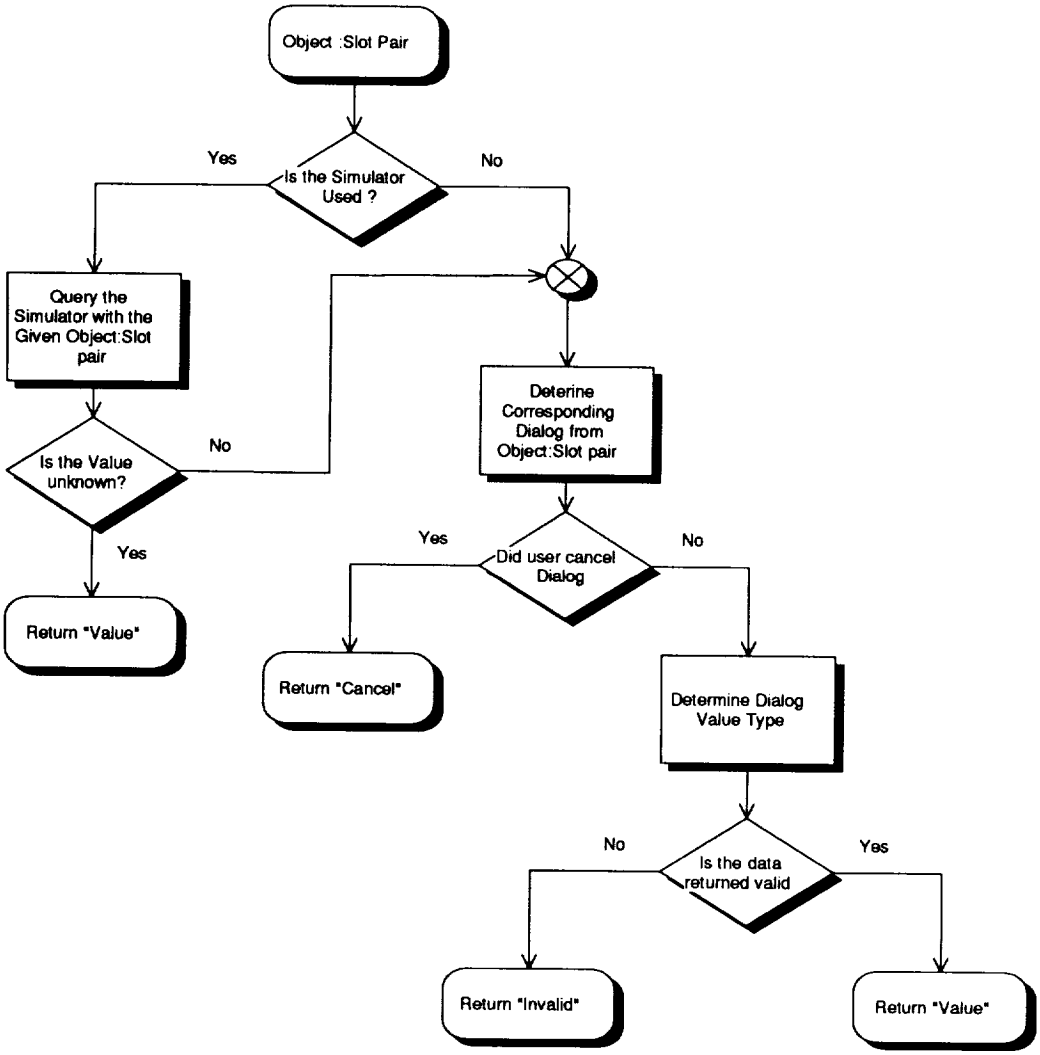

Figure 4.5 -- Data request procedure

An example query follows.

Send-Message (Query-system "Digital-Ground-Terminal Acquired")

Figure 4.5 shows the procedures associated with a data request. The initial procedure determines the simulator status. If the simulator is active, data associated with the object-attribute pair is examined and validation is performed. If data is not obtained, or the simulator is found inactive, the end-user is prompted for the required data. Validation of the data is performed and results conveyed to the proper subsystem.

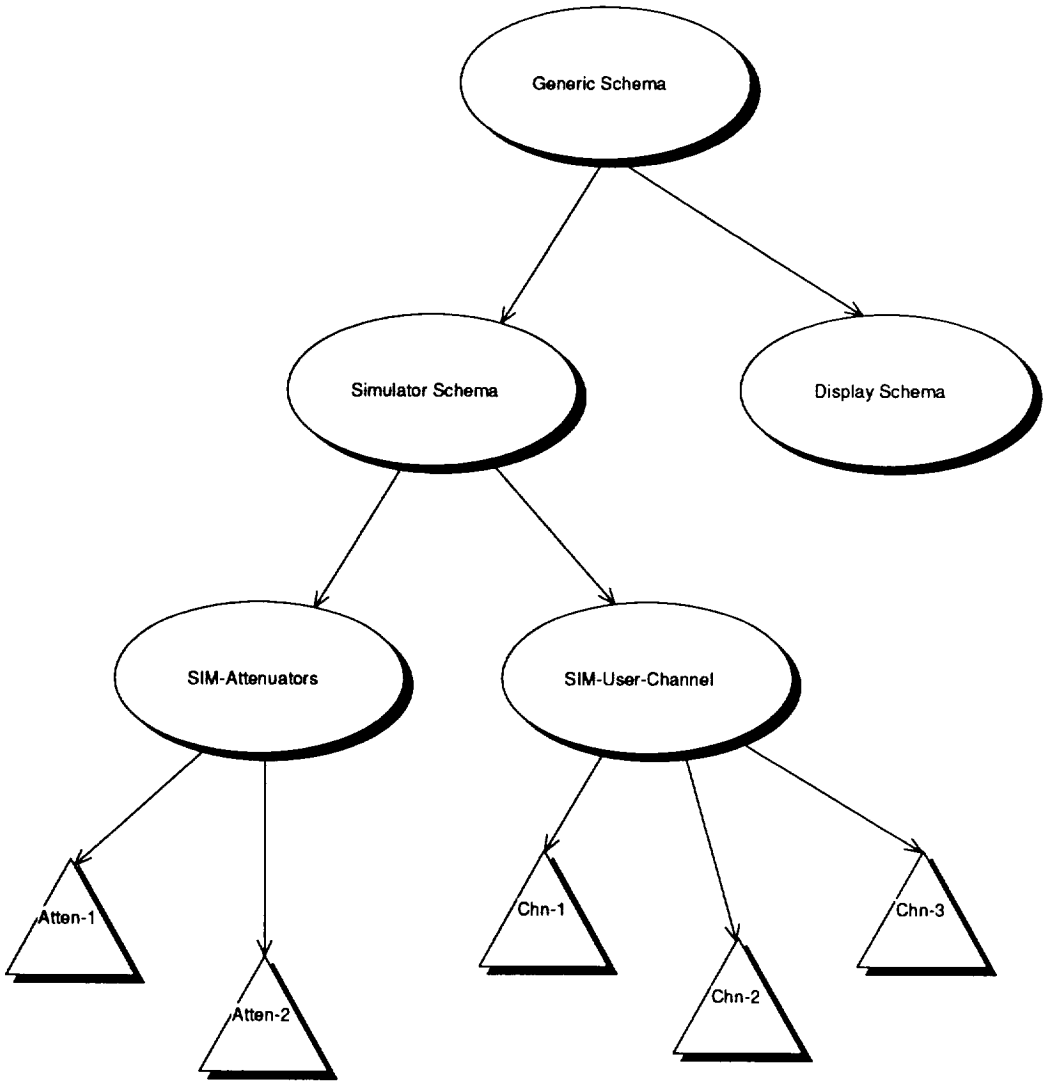

Figure 4.6 -- Simulator Hierarchy

4.4.1 Data simulation

The data simulator was constructed using the schema system defined in the ART-IM environment. The schema system offered a simple mechanism for logically representing the digital ground terminal. The simulator representation of the digital ground terminal is identical to the knowledge base representation. Technically, the knowledge base and simulator schema definitions are not isolated from one another. Therefore, a hierarchical class structure was developed to provide needed isolation between the schemas defined for the simulator and knowledge base systems. This isolation is required so unexpected results do not occur in the execution of the knowledge base. An example of this hierarchy is shown in Figure 4.6.

Since developing a formal simulation of the digital ground terminal was not practical, an ASCII data file is used to initialize the simulator schema descriptions. The format of the data file will be discussed later in this section. Once the simulator has been initialized, access to data is trivial. The message received contains an object-attribute pair corresponding to a ART-IM schema-slot definition. The definition is checked for inclusion in the current simulator hierarchy. If the definition is found, data associated with the schema-slot pair is validated and returned. If the data is unknown data acquisition responsibility is passed to the user dialog (Note: Validation determines only if data exists.).

4.4.2 Simulation data file

The simulator data file is a representation of the objects to be included in the simulation hierarchy. Individual ASCII files can be constructed to represent varying configurations of the digital ground terminal. An example of a simulator file is shown in Figure 4.7. The data file is assembled using the following technique.

Object. Attribute = Value

Acquiring simulation data is initiated by end-user request. Once an end-user requests data from the simulator and the appropriate simulator data file initialized, the simulator becomes active.

The simulator data is initialized by deciphering the named ASCII data file. Data is extracted by parsing the data file to determine the appropriate schema-slot descriptions. These descriptions are compared to the current hierarchy. Schema-slot descriptions missing from the hierarchy, are created dynamically.

The initial values associated with the schema-slot descriptions are obtained and converted to adhering ART-IM data types. Allowable data types are listed in Table 4.4.

Data Type	Example
Integer	256
Hexidecimal	0xBCD
Floating Point	1.0e-3.0
String	"Connected"

Table 4.4 -- Allowable data return types

```
Sim-Transponder.is-a = Simulator-Schema
Sim-Transponder.instance-a = Transponder
Sim-Transponder.Connected = FALSE
Sim-Attenuator.is-a = Simulator-Schema
Sim-Attenuator.is-a = Attenuator
Sim-Attenuator.Connected = FALSE
Sim-Attenuator-1.instance-of = Sim-Attenuator
Sim-Attenuator-1.Setting = 27
Sim-Attenuator-1.Device-Number = 1
Sim-Attenuator-2.instance-of = Sim-Attenuator
Sim-Attenuator-2.Setting = 14
Sim-Attenuator-2.Device-Number = 2
Sim-User-Channel.is-a = Simulator-Schema
Sim-User-Channel.is-a = User-Channel
Sim-Channel-1.instance-of = Sim-User-Channel
Sim-Channel-1.Transmit-Channel = 2
Sim-Channel-1.Receive-Channel = 1
Sim-Channel-1.Data-Rate = 25.0
Sim-Channel-1.BER = 0.000
Sim-Channel-2.instance-of = Sim-User-Channel
Sim-Channel-2.Transmit-Channel = 3
Sim-Channel-2.Receive-Channel = 2
Sim-Channel-2.Data-Rate = 25.0
Sim-Channel-2.BER = 3.000D-5
Sim-Channel-3.instance-of = Sim-User-Channel
Sim-Channel-3.Transmit-Channel = 1
Sim-Channel-3.Receive-Channel = 3
Sim-Channel-3.Data-Rate = 25.0
Sim-Channel-3.BER = 0.0000
Sim-Ground-Terminal-Status.is-a = Simulator-Schema
Sim-Ground-Terminal-Status.is-a = Ground-Terminal-Status
Sim-Ground-Terminal-Status.Acquired = STEADY
```

Figure 4.7 -- Simulator data file

4.4.3 User dialog

An inactive simulator or an unknown simulator value is justification for activating the user dialog sequence. The condition of an inactive simulator is the default condition for data retrieval. The end-user can also explicitly tell GTEX that user dialog is to be employed. A hierarchy, similar to the simulator hierarchy, indicates the correct display dialog to be applied to obtain the necessary data. The display subsystem is responsible for displaying the dialog and returning the entered value. The syntax follows:

send-message display-system "dialog message"

The message is the named dialog display returned by the corresponding dialog schema-slot inquiry.

The query system is also responsible for verifying end-user data. In addition to returning the dialog name, the dialog schema-slot definition returns the symbolic representation of the expected data type. Expected data types are those listed in Table 4.5. The expected data type is compared to the type of data returned by the end-users' response. If the returned data type is invalid, the end-user will be informed and user dialog reiterated. The data is then returned to the appropriate subsystem.

Data Type	Example
Integer	256
Hexidecimal	0xBCD
Floating Point	1.0e-3.0
String	"Connected"

Table 4.5 -- Allowable Data Return Types

4.5 Control system

The control system receives messages from the knowledge base. These messages represent varying conditions of the knowledge base. Depending on the situation, the control system will activate the appropriate push-buttons accordingly. The message syntax is as follows:

Syntax: "Status"
Example: Send-message (control-system "simulator-loaded")

The 'Status' parameter represents the condition of the knowledge base. The following instance represents this type of directed control.

The end-user is given the opportunity to cancel dialog at their discretion. In doing so, the end-user suspends additional rule firings which may occur. The knowledge base is responsible for suspending additional rule-firings and sending a message to the control system that dialog has been canceled. The control system responds by activating the 'Continue' push-button. The end-user then has the ability to; Reset GTEX, Continue Execution or Exit GTEX.

4.5.1 GTEX controls

A mouse pointing device provides control of GTEX execution. This type of device alleviates the end-user from typing commands necessary to perform a task. The first procedure upon entering the control system is checking mouse activity. Detecting none, program execution reverts to the inference engine. Action verification is performed upon detecting mouse activity. Once the action has been verified, an appropriate fact is asserted into working memory. Two types of control are possible: system and display.

4.5.2 System control

System control, represented as push-buttons, are responsible for governing the execution of GTEX. The possible controls are itemized in Table 4.6. Control is provided for selecting an appropriate scenario. Additionally, the capabilities of resetting the system and controlling the inference process are provided. By asserting the appropriate fact the control system informs the knowledge base of the end-users' request.

4.5.3 Display control

Display controls enable the end-user to view varying levels of the digital ground terminal and associated subsystems. If the control system locates mouse activity on the display page, the first procedure determines the current display page. The exact mouse position is then determined. Finding the mouse within valid boundaries, a corresponding fact is asserted into working memory. The knowledge base responds by determining the correct page to display and sending the appropriate messages to the display system.

Optimally, the end-user should notice a physical change to the current display in response to mouse selection. The version of the graphics library used hindered the development of either approach for use with display control. However, unwanted mouse display selections minimally affect the end-user. System control capability allows the end-user to 'back-up' to the next highest system in the hierarchy.

Control Type	Push-Button Name	Description
Display Backup	Backup	Allow end-user to navigate backward through the display hierarchy.
Scenario Display	Scenario	Display available scenarios; A description of each scenario is given.
Select Scenario	Select	Select appropriate scenario.
Verify Fault	Verify	Start the diagnostic process.
Isolate Fault	Isolate	Once verified fault can be isolated.
Recommend Action	Action	Once the fault has been isolated request GTEX to recommend an appropriate action.
Reset System	Reset	Allow the system to be reset; other scenarios can be explored.
Continue Execution	Continue	Allow system to continue from cancellation of dialog.
Exit	Exit	Exit GTEX

Table 4.6 -- System controls and descriptions

4.5.4 Control Realization in the ART-IM Environment

The control system implements a constructor ART-IM defines as an asynchronous function. An asynchronous function provides a mechanism for calling an user defined function automatically between rule firings. Normally, ART-IM suspends execution when there are no more rule activations. A flag included in the ART-IM environment, 'Set-halt-when-no-activations' controls this phenomenon. Setting the 'Set-halt-when-no-activations' flag to NIL insures system execution continues when the agenda is empty. During system execution, once the agenda is empty the system repeatedly calls the predefined

asynchronous function, waiting for end-user interaction. Once the end-user selects one of the available control mechanisms, the system reacts to the fact asserted into working memory. Resetting the 'Set-halt-when-no-activations' flag to T halts system execution.

4.6 Display system

Screen design is still a black art (Peterson 1979). One of the first rules to consider when designing a user interface screen is to keep it as simple and uncluttered as possible. Peterson expresses it in this way.

> "Try to present an entire, logically connected thought on the screen at one time. A good way to tell if you have one idea is, if you can think of a title. If you cannot think of a title for that screen, it probably contains to much data or a bunch of unrelated information."

Ideally, point-and-do interfaces should make clear what properties a screen object has. When the objects on the screen and their associated properties can be inferred, the user then can explore the different screens, acting on individual items as if they were real (Shamonsky 1985). The developer of display screens has to insure that control over data, texts, or formats which are essential to the system operation is not given to the end-user. The user interface must always be under system control. Peterson states this by saying, "Don't give naive or careless users the weapons they need to blow themselves out of the water."

These kinds of situations can be avoided if the end-user is given the opportunity to either back up or back-out safely. This joins with the feature of most point-and-do interfaces, reversibility. Since end-users are given the opportunity to explore the system, allowing for easy reversal of actions, either through an 'undo' option or through complementary opposing actions, can save users considerable aggravation (Schneiderman 1987; Shamonsky 1985).

One other important aspect of screen design is good dialogue. Providing good dialogue means dialogue that is easy to use and is understandable. These are common-sense considerations which must be taken into account. Dialogue as in screen design must be designed to your users' capabilities. By designing the dialogue to your end-users' capabilities, the dialogue will increase the throughput and decrease error rates of the end-user. By providing a dialogue which is not frustrating to the end-user, an improved morale will result. Peterson expresses good dialogue in the following manor;

> "Good dialogue is not easy to achieve. You must understand your users and how they perceive what they do for a living. You must design to your users' jobs. Your users are already out there doing something and they are going to continue to do whatever they do now, only they will be using your latest application to do it."

In conformance with the previous discussion of good dialogue, a program should always respond to the user. Peterson points out that nothing is more frustrating to the user than hitting a key and watching the screen go blank, and not having a clue to the systems' operating status. This can be avoided by giving the end-user a series of in-progress messages. Messages are the system's way of communicating with the user. Peterson cleverly states, "Today's hesitant novice is tomorrows impatient expert."

4.6.1 GTEX graphics interface

Text based graphics displays limit the ART-IM environment. The standard interface tools provided by ART-IM are not capable of handling graphics operations such as lines and circles. ART-IM supports an interface to a graphical set of tools developed by South Mountain Software to construct such complex images. One of the penalties of going with a graphics environment is a slow down in overall system performance; negligible in GTEX.

Encompassing the arguments from the previous discussion and these tools, a graphical environment was developed. The display system of GTEX supports dynamic interaction by the end-user.

4.6.2 Display window

The following categories separates the display window: Display page, Status, and Control areas. The display page and control areas exemplify the point-and-do interface. Using a mouse, the end-user selects active regions for navigation and system control. Status messages keep the end-user informed of system operation.

4.6.3 Message structure

The display system processes several distinct messages from the other subsystems. These message types include Dialog, Display, Initialize, Message, Update, and Warning. These messages provide a modular approach to display design. The display receives messages from the knowledge base and query subsystems.

The general message syntax follows.

```
Send-Message Display-System "Action Action-Message"
```

The 'Dialog' action is responsible for processing system requests for data. The display system upon receiving a dialog action halts system execution. Execution does not continue until the end-user responds to the request. The query subsystem initiates the dialog message.

Syntax: "Dialog Dialog-Name"
Example: send-message (display-system, "Dialog Transponder-Connected")

The 'Display' action permits display of a designated page of information. The associated message indicates the page requested by the end-user. The 'Display' message originates in the knowledge base.

Syntax: "Display Display-Name"
Example: send-message (Display-System, "Display SITE-System-Display")

The 'Initialize' action is a request to place the monitor into the correct video mode and define the display background. The 'Initialize' action is also responsible for sending the correct message for displaying the introduction page. The 'Initialize' action originates in the knowledge base.

Syntax: "Initialize"
Example: send-message (Display-System, "Initialize")

The 'Message' action will display a status message in the status area; discussed later. The status messages allow the end-user to be consciously aware of the current reasoning. The 'Message' action originates in the knowledge base.

Syntax: "Message Message-Number"
Example: send-message (Display-System, "Message 102") [1]

Once the display system receives the 'Update' action, corresponding screen objects are modified. The 'Update' action informs the knowledge base that modifications are complete. It is then the responsibility of the knowledge base to send the appropriate message to update the physical page. The 'Update' action is initiated by the knowledge base.

---

[1] The message number will be discussed later in this section.

Syntax: "Update Device-Change"
Example: send-message (Display-System, "Update no-transponder")

'Warning' actions appear in the form of dialog. This allows dialog to inform the user of the critical conditions about system performance. The 'Warning' action originates in the knowledge base.

Syntax: "Warning Dialog-Name"
Example: send-message (display-system, "Warning Data-unavailable")

Table 4.6 provides a summary of the different actions received by the display subsystem.

Action	Initializing Subsystem	Action Message
Dialog	Query	Dialog-Name
Display	Knowledge Base	Page-Name
Initialize	Knowledge Base	-
Message	Knowledge Base	Message-Number
Update	Knowledge Base	Device-Change
Warning	Knowledge Base	Dialog-Name

Table 4.6 -- Action Messages

4.6.4 Main display

The main display design allows technicians with a minimum amount of training to be proficient in diagnosing the digital ground terminal. The display is dynamic to allow the end-user to pin-point the fault with little effort. This will speed the troubleshooting process. The design guidelines were modeled after the following criteria:

1) Information regarding the status of individual components is known.
2) Providing a method of keeping the display a distinct focus

System specifics are browsed by selecting the appropriate subsystem on the current page. By selecting a subsystem, further detail is revealed. The level of detail included within GTEX allows the end-user to display subsystems to the board-level.

4.6.5 Display page

The end-user can browse the subsystems of the digital ground terminal to pinpoint the fault (if there is one). The end-user will then know where to focus attention in the physical ground terminal for the fault location. Each display page contains the following components:

- Title
- Devices
- Line Segments
- Display Text
- Other

The 'Title' represents the title of the current page. 'Devices' are associated with the different components of the digital ground terminal. Each device corresponds to an individual object or group of objects within the database. Links between devices and page objects are provided by the action messages sent to the display subsystem. This method allows the developer to update the system page independent of the knowledge base. The 'line segments' provide interconnection between the devices. 'Display text' allows miscellaneous text to be presented, This information permits other aspects of the subsystem to be labeled, without creating new devices. 'Other' includes a technique for displaying individual page characteristics not conforming to one of the other categories. This capability provides the flexibility for displaying available values of the varying subsystem components.

4.6.6 Status area

The status area provides the end-user with the current state of the GTEX system execution. Specific messages are relayed to the user which include the following:

- Normal Status
- Request
- Affirmative Status
- Negative Status

Normal Status

Example: Mouse-Handler Installed

The normal status message includes messages representing those situations which occur as in the above example. Included, are all negligible conditions which occur during the execution of the system.

Request

Example: Rqst: Attenuator-1 Setting

Request Messages are displayed when interaction with the simulator or the end-user is required. This provides a resource for the end-user to perceive what data is being referenced.

Affirmative Status

Example: Ground Terminal Isolated

Webster's Dictionary includes the following definitions of affirmative and positive.

Affirmative:	Asserting that a fact is so.
Positive:	Affirming the presence of that sought or suspected to be present.

Relating this to GTEX, affirmative status displays facts about the digital ground terminal that are found to have occurred in a favorable manner.

Negative Status

Example: Transponder NOT Connected

Webster's Dictionary includes the following definition of negative.

Negative:	Not affirming the presence of the organism or condition in question.

Negative status messages illustrate facts about the digital ground terminal which cannot be found affirmative.

As noted, the syntax of the message action includes a message number. This message number corresponds to the status message to be displayed. Upon entering GTEX, a message database is initialized. The numeric delineation

provides an indexing strategy for the database. Table 4.7 lists the numeric range representing the status messages categories.

Status message classification also permits a color scheme display representation. The color scheme is defined in Table 4.8. Color offers an efficient means of representing the types of status messages. The end-users can quickly scan and decipher messages, allowing the current line of investigation to be determined.

Numeric Range	Status Type
0 - 99	Normal Status
100-199	Request Status
200-299	Affirmative Status
300-399	Negative Status

Table 4.7 -- Numeric Range of Status Messages

Color	Status Type
Brown	Normal Status
Cyan	Request Status
Green	Affirmative Status
Red	Negative Status

Table 4.8 -- Color Representation of Status Message

The display system handles status message requests by performing a database search. The database is parsed to determine the correlation of the message position versus the numeric representation. The results are accumulated into a linear array. Indexing of the array is based on the numeric representation of the status message. Keeping memory requirements to a minimum, only the file record position is stored. When a request is made the status message number is determined and appropriate record number retrieved. The database is queried and extracted status messages displayed.

Status messages are represented using two display formats. The active message is displayed in a highlight or bold style. This gives the end-user a quick reference in locating the current message. Additionally, the status area retains the previous eight messages. End-users see a history of messages, which assist in deducing the current reasoning process. These additional messages are displayed in the normal intensity of color corresponding to the status type.

4.6.7 Control area

The control area provides the visual reference for controlling the system. The controls are implemented as push-buttons. See the section on the control system in this chapter for a description.

4.6.8 User dialog

Two methods are implemented for obtaining data in GTEX; user dialog and simulation inquiry. By providing user-dialog, the system can prompt for needed information. Besides providing needed information, dialog can be used to convey important messages likely missed in the status message area. Keeping consistent with the graphical user interface, the ART-IM capabilities of supporting user-dialog where abandoned. Implementing the ART-IM style of user-dialog meant switching between different display modes which was not feasible. Therefore, a scheme was constructed for supplying user dialog. User-dialog is completed synchronously, requiring end-user response for continuation of system execution. Figure 4.8 is a representation of the different types of dialogs provided by GTEX.

Several tools created allow the end-user to respond to given dialog. These tools include push-buttons, radio-buttons and edit-text.

Push buttons serve a dual role in the dialog scheme. They are responsible for either gathering information as in yes or no situations, or to provide control over data provided by the other tools. By selecting a push-button the dialog box is removed and data recorded.

Radio-buttons limit the possible responses of the end-user to the given dialog. This type of dialog is useful when limited choices for a given dialog are available. Radio buttons operate in an exclusive-or fashion. Multiple radio button groups can be included in a dialog at the developers' discretion.

Edit-text provides the user with an opportunity to type in the values requested by the dialog. Editing capability allows the end-user to respond when data is unbounded. One disadvantage of giving the end-user editing capability is providing an opportunity for him to make mistakes when entering data.[2] Therefore, using editing capability should be considered last when developing user-dialog.

---

[2]Note; The user dialogue provides a vehicle which to request information, the query system is responsible for checking the integrity of the actual values.

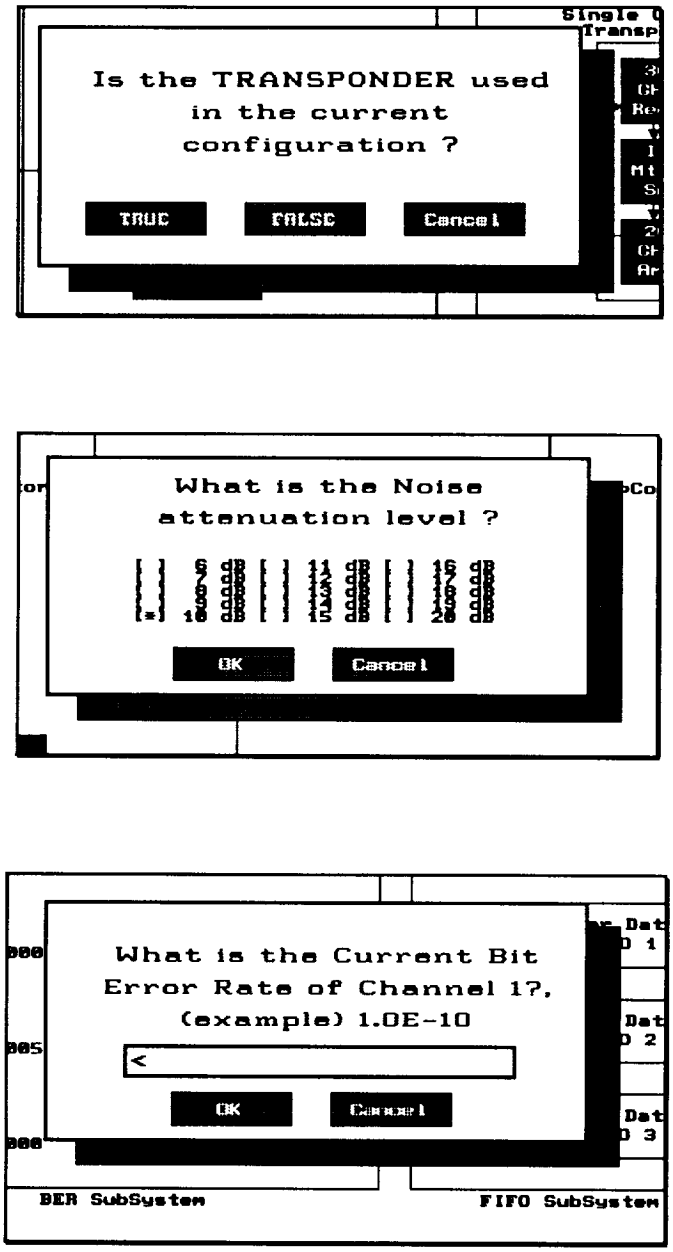

Figure 4.8 -- GTEX Dialog Displays

CHAPTER V

OBJECT-ORIENTED PROGRAMMING

5.1 Introduction

As humans it is more natural for us to think about our world in terms of different objects. Consider an automobile; if someone was asked, "What is an automobile," they might respond by saying that an automobile is a means of transportation from destination-A to destination-B. They may also add, "To get from destination-A to destination-B the automobile requires sufficient amounts of different fluids that allow the automobile to operate."

One classification of the automobile is that it is a mode of transportation. If asked to describe our automobile, we could respond by giving the general characteristics of an automobile, saying that it has: four tires, either two or four doors, a steering wheel, an engine, etc. We could also describe our automobile by stating the make, model and year it was assembled. These characterizations allow us to define other classifications of our automobile.

This idea of objects is extending into the way computer programs are developed. This programming philosophy is termed object-oriented programming. One author describes object-oriented programming in the following way.

> "Object-oriented development is fundamentally a new way of thinking and not a programming technique. The development of an object-oriented application is a conceptual process independent of a programming language until the final stages. Superficially the term 'object-oriented' means that we organize software as a collection of discrete objects that incorporate both data structure and behavior. This contrasts with conventional programming in which data structure and behavior are only loosely connected." (Rumbaugh, et al. 1991)

5.2 Characteristics of an object

The two important building blocks of object-oriented technology are data abstraction and encapsulation (Rumbaugh, et al. 1991).

Abstraction is defined as, "a mental facility that permits humans to view real-world problems with varying degrees of detail depending on the current context of the problem" (Rumbaugh, et al. 1991). Abstraction focuses on the general characteristics of an object. Emphasis during system development is placed on what the object is and does, before deciding how it is to be implemented. As in the example of the automobile, the characteristics that are common to all automobiles; four tires, an engine, how to operate the automobile, etc, become an abstraction of the group of individual automobiles. Proper abstraction allows our model to be used from the initial design to the actual implementation.

Encapsulation, or information hiding, is a technique the divides the external characteristics of an object from the internal, implementation details of the object (Rumbaugh, et al. 1991). By applying encapsulation principles to objects, each object's implementation is independent. This prevents a program from showing signs of the ripple effect, where small changes have massive side-effects. Therefore, the implementation of an object can change without affecting the other objects addressing it.

By using abstraction and encapsulation as building blocks the following characteristics of object-oriented programming evolve: identity, classification, polymorphism and inheritance.

5.2.1 Identity

Objects can either be tangible like a digital earth station, or conceptual such as a communication signal from the earth station to the satellite. Identity implies that individual objects have a distinct personality. The concept of identity also applies to multiple objects having identical characteristics. Consider two automobiles having identical features; each automobile is constructed individually with distinct components, giving each its own identity.

5.2.2 Classification

A classification or 'class,' is constructed by applying abstraction principles to a given set of objects. A class can be thought of as a template, from which individual objects are cloned. A class definition incorporates those characteristics and behaviors common among the set of objects. The characteristics chosen are arbitrary and depend on the application at hand.

5.2.3 Polymorphism

Operations, called 'methods,' are bound to individual classes. Methods describe the general behavior exhibited by an individual class. "Polymorphism implies that the same operation can behave differently on different classes (Rumbaugh, et al. 1991)."

This concept is best described by an example. Consider the following:

> A computer card-game with graphics was designed using object-oriented techniques. The developer defined classes representing both the individual players and a graphics display window. A operation called 'draw' was applied to both classes. The behavior of the 'draw' operation of the card-game class represents the act of drawing a card from a deck. The behavior of the 'draw' operation of the display window class describes the steps required to display a graphics window.

This computer card-game exhibits polymorphic behavior. The behavior of the 'draw' method changes, when associated with the different classes.

5.2.4 Inheritance

Inheritance is the sharing of characteristics and behaviors among objects based on a hierarchical relationship (Rumbaugh, et al. 1991). By applying abstraction to sets of classes, a general class or 'superclass' evolves. Each member of the set becomes a subclass of this new superclass. The new subclasses will inherit those characteristics and behaviors now displayed by the superclass and retain those features that make it unique.

5.3 Object-Oriented Techniques in GTEX

Object-oriented techniques are used extensively throughout GTEX. The techniques used in the development of the GTEX modules will be described. The following classifications are used.

- Ground Terminal Model
- Display Hierarchy
- Display Objects

5.3.1 Earth Station Model

The characteristics of the digital earth station are represented in GTEX by an object hierarchy. A generic object representation of the earth station is shown in

Figure 5.1. Three classifications of the earth station were defined: knowledge base, dialog, simulator. The object hierarchies were developed using the ART-IM schema notation. A prefix notation was used in defining each classification. The prefixes are; 'KB-', 'DLG-', and 'SIM-' respectively.

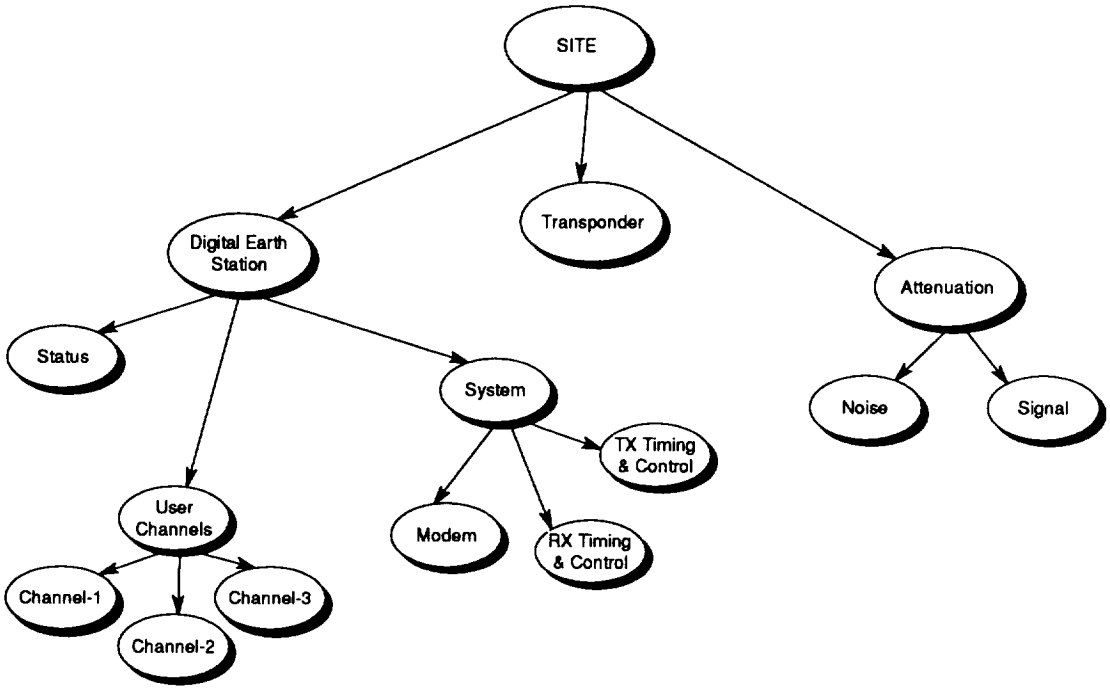

Figure 5.1 -- Earth station hierarchy

The knowledge-base hierarchy provides the model for reasoning about the earth station. Figure 5.2.b illustrates an example of a knowledge-base schema.

The dialog hierarchy was created to provide an efficient means of retrieving dialog information. The dialog hierarchy is identical to the knowledge base hierarchy except the attribute values represent the dialog message responsible for obtaining the necessary piece of data. Figure 5.2.c illustrates an example of a dialog schema.

The simulator hierarchy provides a third model of the earth station. The simulator hierarchy represents a physical earth station configuration. This information is queried as if GTEX was physically connected to the earth station. Figure 5.2.d illustrates an example of a simulator schema.

```
(defschema User-Channel
    (Configured FALSE)
    (Transmit-Channel UNKNOWN)
    (Receive-Channel UNKNOWN)
    (Data-Rate UNKNOWN)
    (Channel-Number UNKNOWN)
    (BER UNKNOWN)
    (Error-Channel UNKNOWN)
    (tx-rqst-msg UNKNOWN)
    (rx-rqst-msg UNKNOWN)
    (dr-rqst-msg UNKNOWN)
    (error-msg UNKNOWN)
    (valid-msg UNKNOWN))
```

(a)

```
(DEFSCHEMA KB-Channel-1
    (INSTANCE-OF KB-User-Channel)
    (Channel-Number 1)
    (tx-rqst-msg 206)
    (rx-rqst-msg 207)
    (dr-rqst-msg 208)
    (ber-rqst-msg 215)
    (error-msg 308)
    (valid-msg 111))
```

(b)

```
(DEFSCHEMA Dlg-Channel-1
    (INSTANCE-OF Dlg-User-Channel)
    (Channel-Number 1)
    (Transmit-Channel ("TX-Channel-1" INTEGER))
    (Receive-Channel ("RX-Channel-1" INTEGER))
    (Data-Rate ("DT-Channel-1" FLOAT))
    (BER ("BER-Channel-1" FLOAT)))
```

(c)

```
(Schema SIM-Channel-1
    (INSTANCE-OF SIM-User-Channel)
    (Channel-Numer 1)
    (Transmit-Channel 2)
    (Receive-Channel 1)
    (Data-Rate 25.0)
    (BER 1e-10))
```

(d)

Figure 5.2 - Schema definitions in GTEX
(a) Generic Definition (b) Knowledge Base Schema
(c) Dialog Schema (d) Simulator Schema

5.3.2 Display Screen Hierarchy

Since the end-user of GTEX is capable of navigating through several screens, an interconnection between the varying screens was required. A hierarchical relationship was developed. The screen hierarchy is shown in Figure 5.3.

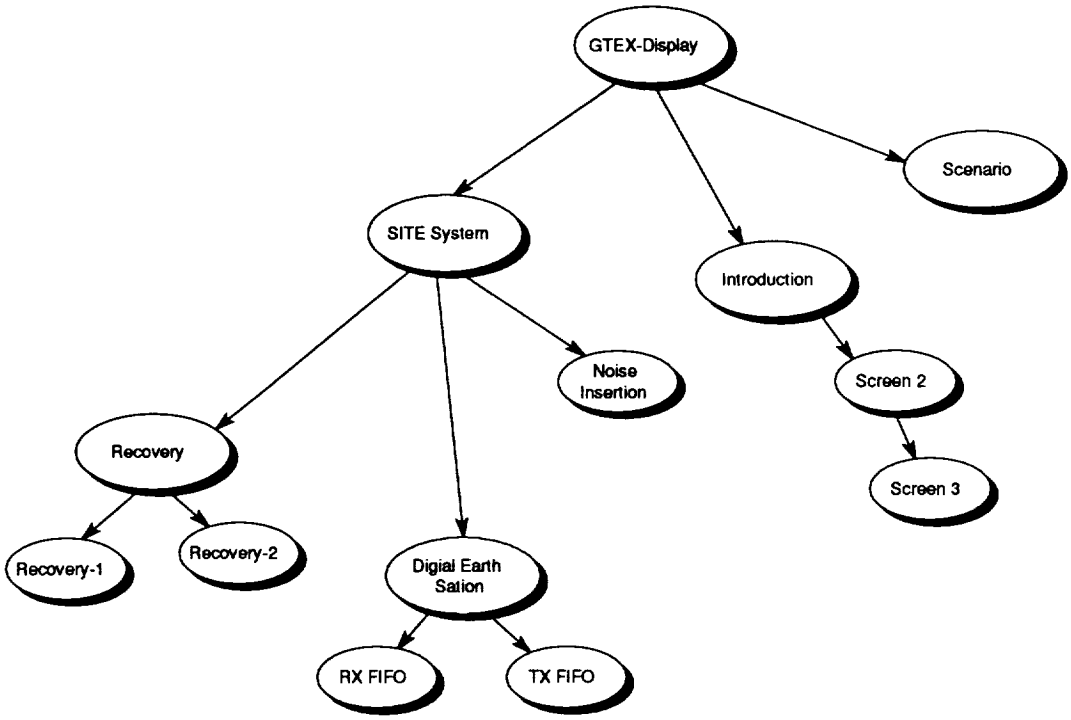

Figure 5.3 -- GTEX screen hierarchy

The display screen hierarchy is used by the knowledge base and control modules. The knowledge base is responsible for setting the current page based on working memory conditions. The control system uses the hierarchy to determine which areas of the current page are active for navigation. The display hierarchy was developed using the ART-IM schema notation.

5.3.3 Display Page Objects

A technique was needed to provide an efficient means of constructing the necessary dialog and displays screens. Representing these entities as objects proved to be the most effective. Table 5.1 lists the various classes available. Of the available classes, two are dominant, the window and dialog.

Object	Attributes	Methods
Point	X-Coordinate Y-Coordinate	
Dimension	Length Width	
Button	Point Dimension Status Active Label Fact Asserted	Up Down Action when Selected Pressed Released Initialized Valid
ButtonGroup	Array of Button	
Dialog Text	Point Color Text String Alignment Font	Initialize
Edit Dialog	Point Dimension Active Text String	Initialize Print Text
RadioButtonGroup	ButtonGroup OldButton NewButton	Initialize Update
RadioButtonSet	Array of RadioButtonGroup Active	Initialize

Table 5.1.a - Display Objects

Object	Attributes	Methods
Dialog	Point Dimension DialogText ButtonGroup EditDialog RadioButtonSet	Initialize
Device Text	Point Color Text String Alignment	Draw
Device	Point Dimension Color Device Text	Draw
Line Segment	Point Length Direction Pointer	Draw
Display Text	Point Color Alignment Font Text String	Draw
Window	Title Array of Device Array of Line Segments Array of Display Text	Draw Draw Misc.
MouseButton	Left Right	
Mouse	Point MouseButton	Pressed Released

Table 5.1.b -- Display Objects Continued

Figure 5.4 shows the hierarchy of the window class. As illustrated, devices, line segments and display text provide the foundation of the display window. Active areas on each window were created by combining the device and button classes.

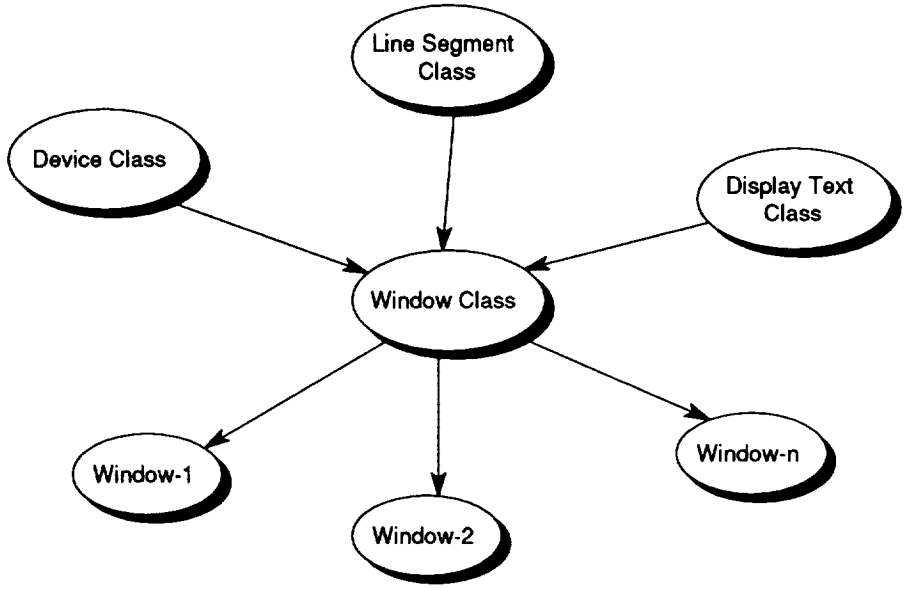

Figure 5.4 -- Window hierarchy

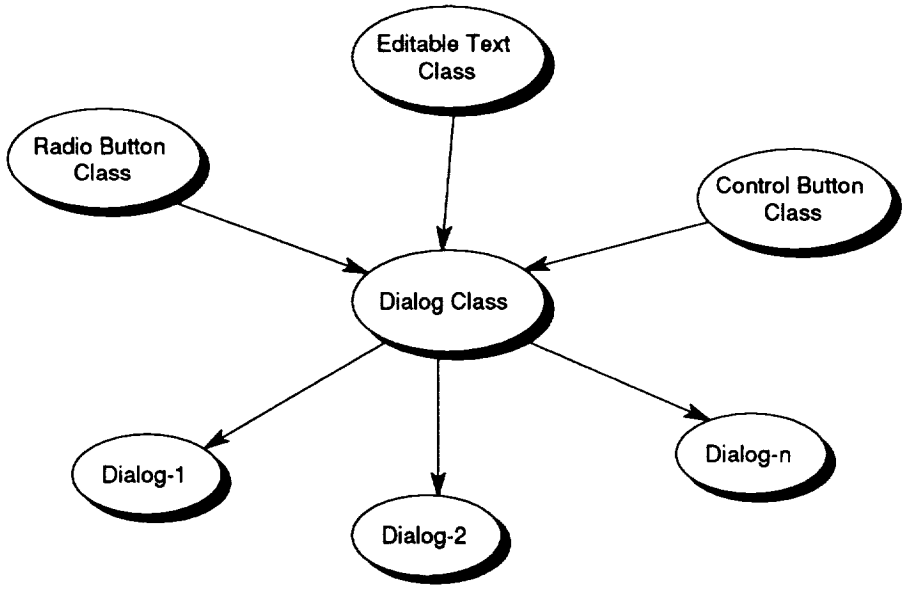

Figure 5.5 -- Dialog Hierarchy

Figure 5.5 shows the dialog class hierarchy. The dialog is able to obtain information from either control buttons, radio-buttons and editable text. The display objects' representations were created using 'C' language programming techniques.

5.4 Extending the Object Capability of GTEX

Variations on the C programming language were used to simulate the objects needed by the display system. The concept of the structure in the C language was used for the representation of the class; example is shown in Figure 5.6.a.

The use of a constructor in the 'C' language called a "structure," allows complex data to be represented. The 'C' language allows arrays of structures, structures to be nested, and the address of a structure to be determined. These characteristics provided the basis for simulating a class hierarchy. Other structure members may include other valid data types, giving the object its characteristics, etc. A typical instance declaration is shown in Figure 5.6.b.

```
struct Window
{
        char Title[40];
        struct Device *WinDevice[20];
        struct LineSegment *WinLineSeg;
        struct DisplayText *WinDspTxt;
        void (*DrawMisc)();
};
```

(a)

```
struct Window wnGndTrmSub =
{
        "Phase II - Digital Ground Terminal",
        { &dvTxFIFO11, &dvTxFIFO21, &dvTxFIFO31, &dvTxScrambler1,
          &dvTxOrderwire1, &dvParToSerial1, &dvModulator1, &dvDemodulator1,
          &dvSerialToPar1, &dvStrecher1, &dvRxReference1,
          &dvRxFIFO11, &dvRxFIFO21, &dvRxFIFO31,
          &dvTxTiming1, &dvRxTiming1, &dvOWProcessor1, (struct Device *) NULL},
        &lsGndTrmSub[0],
        &dtGndTrmSub[0],
        vfnGndTrmKey
};
```

(b)

Figure 5.6 -- 'C' Language representation of objects
(a) Class definition (b) Instance declaration

Gaining the address of the structure was vital. This allows the structure to be passed by reference instead of passing individual structure members. It was also important in defining class methods. Referencing a structure by address allowed functions to be generically created. The generic functions provided the concept of a method. Methods are referenced as a structure member. Since each structure can have the same member name as another structure, polymorphic relationship exists between classes. Figure 5.7 shows a typical method in the display environment.

```
void    vfnDrawWindow( struct Window *wnWinDef)
{
        int iLine;
        struct LineSegment *TempLineSeg;
        struct DisplayText *TempDspTxt;

        m_curoff();

        vfnClearScreen();

        vfnPrintTitle(wnWinDef->Title);

        iLine = 0;

        while(wnWinDef->WinDevice[iLine] != NULL)
                dmdDevice.vfnDraw(wnWinDef->WinDevice[iLine++]);

        TempLineSeg = wnWinDef->WinLineSeg;

        while( (wnWinDef->WinLineSeg != NULL) &&
                        !wnWinDef->WinLineSeg->bEndFlag )
                                lsmdLine.vfnDraw(wnWinDef->WinLineSeg++);

        wnWinDef->WinLineSeg = TempLineSeg;
        TempDspTxt = wnWinDef->WinDspTxt;

        while( (wnWinDef->WinDspTxt != NULL) &&
                        !wnWinDef->WinDspTxt->bEndFlag )
                                dtmdText.vfnDraw(wnWinDef->WinDspTxt++);

        wnWinDef->WinDspTxt  = TempDspTxt;

        wnWinDef->DrawMisc();

        m_curon();
}
```

Figure 5.7 -- Object method in 'C'

5.5 The C - Language Naming Convention

A variable naming convention was defined to enhance debugging ability and understandability of the 'C' code. The Hungarian naming convention was used. This convention defines how variables are created. Very simply, the variable name begins with a lower-case letter or letters that denote the data type of the variable. This notation, adopted by the Microsoft Windows programmers, helps to avoid errors in the code before they turn into bugs. Because the name of a variable describes the use of the variable and its data type, there is less chance of coding errors involving mismatched data types. Table 5.2 is a listing and description of the prefixes used in the coding of GTEX.

Prefix	Variable Type
ao	Art Object
aofn	Art Object Function
as	Art Symbol
at	Art Template
b	Boolean
ba	ButtonAction
bf	Buffer
bfn	Boolean Function
bg	ButtonGroup
bs	ButtonStatus
bt	ButtonType
da	DisplayAction
dafn	DisplayAction Function
dg	Dialog
dm	Dimension
dm	DisplayMessage
dmd	Device Method
dt	Device Text
dv	Device
ed	EditDialog
f	File
g	Graphics buffer
i	Integer

Prefix	Variable Type
ls	Line Segment
lsmd	Line Segment Method
mb	Mouse Button
mbp	Mouse Button Pointer
mgt	Message Type
mgtfn	Message Type Function
mr	Message Record
mt	Mouse Type
pt	Point
rbg	Radio Button Group
rbs	Radio Button Set
s	String
sfn	String Function
sy	System
syfn	System Function
u	Unsigned
ud	UpdateDevice
ufn	Unsigned Function
v	Void
vfn	Void Function
wm	Window Method
wn	Window

Table 5.2 -- Prefix definitions

CHAPTER VII

SUMMARY

The GTEX prototype was successfully completed. The GTEX prototype demonstrates the feasibility of applying expert system technology to the area of satellite communications. The modular architecture of GTEX reduced the complexity of the development environment. Allowing each system to function independently, eases future system development. This modularity also allows GTEX to be ported to other platforms.

By incorporating object-oriented techniques in the GTEX design, maintainability of the individual systems increased. It also increased the functionality of the individual systems. This increase was significantly noticed in the development of the display system. By applying object-oriented techniques the display system became more dynamic.

Implementing rule classifications increased the functionality of the knowledge base. The display rules allowed the knowledge base to control graphics, adding to the dynamic nature of the display environment. Demon rules increased the functionality of the ART-IM object system by allowing changes in working memory to affect the system operation.

Creating a query system capable of accessing various data sources a flexible system was established. The query system supplies data after receiving an inquire from the knowledge base thus, the knowledge base is unaware of how the data originated. The current framework of the query system provides the necessary structure for creating a dynamic link with the earth station.

Future considerations include increasing the capability of the current knowledge base and establishing a physical connection with the current earth station for data retrevial. This enhanced capability will increase the opportunity for transferring this technology to the public sector.

REFERENCE / BIBLIOGRAPHY

ART-IM Programming Language Reference, Inference Corporation, Los Angles, California, 1989.

ART-IM in the DOS Environment, Inference Corporation, Los Angles, California, 1989.

ART-IM Reference Manual, Inference Corporation, Los Angles, California, 1989.

Budinger, J., "A Burst Compression and Expansion Technique for Variable-Rate Users in Satellite-Switched TDMA Networks," NASA Technical Memorandum 102414, Cleveland, Ohio, January, 1990.

Bulman, D., "An Object-Based Development Model," Computer Language, August 1989, 49-59.

Chase, S., "VSATs in America Who's Going to Survive?," Via Satellite, November 1990, 40-48.

Harris, L., "User Interfaces for Inference-Based Programs," AI Expert, October 1990, 42-46.

Ivanic, W., Andro, M., Nagy, L., Budinger, J., Shalkhauser, M., "Satellite-Matrix-Switched, Time-Division-Multiple-Access Network Simulator," NASA Technical Paper 2944, Cleveland, Ohio, October, 1989.

Levenberg, J., "How much does a VSAT Network Cost," Via Satellite, February, 1992, 62-64.

Martain, J., Oxman, S., Building Expert Systems A Tutorial, Englewood Cliffs, NJ, Prentice Hall, 1988.

Peterson, D., "Screen Design Guidelines," In Tutorial: End User Facilities in the 1980's, editied by James A. Larson, NY,IEEE, 1982.

Petzold, C., Programming Windows, Redmond, Washington, Microsoft Press, 1990.

Rolston, D., Principles of Artifgical Intelligence and Expert Systems Development, New York, NY, McGraw-Hill, 1988.

Rumbaugh, R., Blaha, M., Premerlani, W., Eddy, F. Loreson, W., Object-Oriented Modeling and Design, Englewood Cliffs, NJ, Prentice Hall, 1991.

Schlegelmilch, R., Durkin, J. Petrik, E., "GTEX: An Expert System for Diagnosing Faults in Satellite Ground Stations," Proceeding of the Space Communications Technology Conference Onboard Switching and Processing, November 12-14, 1991, Cleveland, Ohio, 103-12.

Shalkhauser, M., "Satellite Ground-Terminal User Simulation," NASA Technical Memorandum 100234, Cleveland, Ohio, January, 1988.

Shalkhauser, M., "Design and Implementation of a Microcomputer Based User Interface Controller for Bursted Data Communications Satellite ground Terminals," NASA Technical Memorandum 101375, Cleveland, Ohio, December, 1988.

Sinha, A., Agrawal, B., Wu, W., " Trends in Satellite Communications Technology, Techniques and Applications," International Journal of Satellite Communications. Vol8, February, 1990, 283-294.

Smith, R., The Facts on File Dictionary of Artificial Intelligence, New York, NY, Facts on File, 1989.

Stipp, L., Kowalski, B., "Object Processing for Knowledge-Based Systems," AI Expert, October 1990, 34-41.

Waite, M., Prata, S, Martin, D., The Waite Group's C Primer Plus User-Friendly Guide to the C Programming Language, Indianapolis, Indiana, Howard W. Sams & Company, 1988.

Waterman, D., A Guide to Expert Systems, Reading, Pa, Addison-Wesly, 1986.

Windmiller, M., "Unique Bit-Error-Rate Measurement System for Satellite Communication Systems," NASA Technical Paper 2699, Cleveland, Ohio, March, 1987.

APPENDIX

APPENDIX 1

MODIFYING GTEX

I. Required software:

Microsoft C version 5.1 - Microsoft Corporation
ART-IM version 2.1 - Inference Corporation
Essential Graphics Library 3.0 - South Mountain Software
DOS 3.3 or higher

Provided is a brief discription of the required procedures needed for making modifications to the GTEX system. A strong background programming in 'C' is recommended.

II. Modifying the knowledge base

Files required: *.ART"

1. These follow the procedures as outlined in the ART-IM manual for creating rules, objects, ART-IM functions, etc.

2. The rule classification and categories discussed in this report should be used.

3. The GTEXD.EXE is the development environment used in debugging and construction. This is the ART-IM development studio.

4. The contents of the files are loaded into the environment by typing (load "ld-gtex.art") in the command window of GTEXD

5. The EMACS editor was used in creating these files. ART-IM provides a editor within the studio that can also be used.

III. Creating a new screen

Files required: DPSCHEMA.ART
DSPDEF.C
DISPLAY.C
CNTLDEF.C
DSPDEF.H

1. Source code for all screens is located in DSPDEF.C.

2. To create a new display a new structure of the type Window needs to be defined, the structure definition can be referenced in DISPLAY.H

3. Once the screen has been defined, the function 'aofnSetDisplay()' in DISPLAY.C needs to be modifies to indicate the addition of the screen.

4. Next is to make a definition reference for the new display, this is completed by appending a quoted string to the 'sMessageActions' array. This array is the message that is sent by the knowledge base for that screen. A typedef array is associated with each reference, 'msgact.' The two arrays must correspond or the wrong screen will be displayed. The variable 'uNMesAct' needs to be updated to indicate the number of messages.

5. Since the reference of the screen is located in another file, the DSPDEF.H is used for making the external reference to the newly created display.

6. Since a new screen was created, the schema hierarchy must reflect this change. The display schemas are located in DPSCHEMA.ART

7. If active areas are to be include, the file CNTLDEF.C has to be modified.

8. A button group of the available regions needs to be created. If, not active regions are to appear then the button group is initialized to NULL.

9. Each button structure is used to define the active regions on the display. The convention used was to overly buttons onto devices.

10. The DSPDEF.C file contains an array 'wnList' associates the displays with the active region representation.

11. The files now need to be complied. This can be performed using the procedure described in the Compilation section.

IV. Creating an update for a screen

Files required: DSPDEF.C
DISPLAY.C
DSPDEF.H

1. In the file DPSDEF.C an Update structure needs to be created, Update structures are only used for devices. The Update structure is an array of device position in the Window structure and the pointer to the new device to be displayed.

2. The DSPDEF.H is used for declaring the external reference to the Update structures defined.

3. The function 'aofnUpdate' in DISPLAY.C needs to be modified to indicate the message the update will activate on. Arrays similar to the display arrays, 'sUpdateMessage' and 'UpdateMessage,' need to be modified. The variable 'NUPMESG' indicates the number of messages.

4. The files are compiled using the procedure are described in the Compilation section.

V. Creating a new dialog

Files required: DGSCHEMA.ART
DISPLAY.C
DLGDEF.C
DLGDEF.H

1. To create a new dialog, modify the file DLGDEF.C.

2. A structure definition Dialog is used in creating a new dialog, The definition can be found in DIALOG.H.

3. Several generic button groups have been defined, located at the top of the file DLGDEF.C. This cuts down on the amount of code that is needed. These should be used whenever possible.

3. After the dialog has been created the reference to this dialog is located in DLGDEF.H

4. If the dialog is returning data then a dialog should be created as a dialog message. An array exist in the DISPLAY.C file called 'sDialogActions' contains the reference to the various dialogs. The array 'dgDialogActions,' is a corresponing array with pointers to the various dialogs. NDLGACT indicates the number of dialog messages.

5. If the dialog is just some kind of message, the dialog should be created as a warning message. Similar arrays exists of the warning messages, 'sWarningActions,' 'dgWarningActions.' NWARN indicated the number of warning messages.

6. The files are compiled using the procedure described below.

7. If the dialog is to retrieve a piece of data, then the corresponding object:attribute pair in the dialog hierarchy needs as its value the name of the dialog that was given as the message reference.

VI. Compilation

Files required: GTEX
GTEX.LNK

1. If the developer is unfamiliar with a MAKE procedure, it is used to ease the compilation process.

2. The GTEX and GTEX.LNK files may require some modification to reflect to location of the :

C compiler and libraries
ART-IM libraries
Essential Graphics libraries
GTEX source code

3. A new development environment can be created by entering MAKE GTEX and the command line in DOS.

APPENDIX 2

FILE DESCRIPTIONS

File Name	Description
GTEX.EXE	Main program
GTEX.MSG	Database of messages
SATDISH.PCC	Bit file of satellite picture on introduction screen
SIM-1.DAT	Simulator data file - Scenario 1
SIM-2.DAT	Simulator data file - Scenario 2
Directory ART-SRC	
ARTFUN.ART	ART-IM function definitions
DGSCHEMA.ART	Dialog schema definitions
DPSCHEMA.ART	Display schema definitions
GLOBAL.ART	Global declarations
GNSCHEMA.ART	Generic schema definitions
INITIAL.ART	Initialization rules
ISOLATE.ART	Isolation rules
KBSCHEMA.ART	Knowledge base schema definitions
LD-GTEX.ART	Loads all files into development environment
MISC.ART	Miscellaneous rules
RECOMEND.ART	Recovery recommendation rules
SOURCE.ART	Data Source rules
USERFUN.ART	User function declarations
VERIFY.ART	Verification rules
Directory C-SRC	
CNTLDEF.C	Active display region definitions
DISPLAY.C	Display message definitions
DLGDEF.C	Dialog definitions
DSPDEF.C	Display screen definitions

File Name	Description
Directory C-SRC continued	
DSPUTIL.C	Display utility functions
GEN_MES.C	Database message generation function
GTEX.C	Main program
INITFUNS.C	ART-IM generated, user function definitions
INITLIBS.C	ART-IM generated
INITSIM.C	Simulator definition function
MSDIALOG.C	Dialog functions
MSHANDLE.C	Control functions
QUERY.C	Query Subsystem functions
SHADE.C	Shading definitions
UTILITY.C	Miscellaneous functions
Directory Development	
GEN_MESS.XE	Database message executable
GTEXD.XE	Main development executable
MESSAGE.TX	ASCII text of messages
Directory FONTS	
BOCKLIN.SET	Bocklin font
MISCREANT.FY	Micro print font
ROMAN25.FY	Roman 25 font
SBO.FY	SBO font
Directory INCLUDE	
COLOR.H	Color definitions
DIALOG.H	Dialog class definitions
DISPLAY.H	Display class definitions
DLGDEF.H	Reference to dialogs
DSPDEF.H	Reference to displays
FONTS.H	Font definitions
GTEX.H	ART-IM definitions
INITFUNS.H	ART-IM generated - user functions
PCX.H	PCX structure definition
UTILITY.H	Miscellaneous function definitions

File Name	Description
Directory MAKE	
GTEX	Make file for compilation
GTEX.LNK	Link file for compilation