DRAFT

The
Initial Academy Training System
Voice Communication System
Guidance Document

December 22, 2004

Status: Final

Preface

TEMPLATE CHANGE LOG

Submission	Date
Preliminary	June 9, 2004
Draft	September 17, 2004
Final	December 22, 2004

Table of Contents

1 Engineering Team

Below is the list of engineering team members:

Development Team			
Stephen Souder	Lead Engineer	609-485-6170	ACB-230 / EIIF
Zack Bocelle	Client Software POC	609-485-8761	ACB-230/EIIF
Bill Pfeiffer	COM Server POC	609-485-7906	Enroute Computer Solutions/ EIIF
Michael Perseo	DSP POC	609-485-7915	Joseph Sheairs Associates/ EIIF
Steve Bakanas	DSP POC	609-485-7916	Joseph Sheairs Associates/ EIIF
Tom MacWright	VCS Adaptation Builder POC	609-485-6170	Joseph Sheairs Associates/ EIIF
John Gauntt	VCS Hardware POC	609-485-8075	J&N/EIIF

2 Reference Documents

MSDN Library for Visual C++ and Visual Basic Version 6.0

V++ User Manual

Impulse Studio ActiveX Components Reference

SMx Audio System User Manual

Farpoint Objx ActiveX Components Reference

GMS ActiveX Components Reference

Impulse Studio ActiveX Components Reference

VSCS User's Guide

3 Purpose

This document is intended to provide guidance for continued support of the Initial Academy Training System (IATS) Voice Communication System (VCS), which was initially developed at the EnRoute Integration and Interoperability Facility (EIIF), located at the William J Hughes Technical Center (WJHTC) in Atlantic City, NJ.

4 Definition of Terms

The following list is a definition of terms provided for use in the context of this report

- **Graphical User Interface** – The common practice of providing a graphic (i.e. visual object) to a user for interaction with a software application.
- **Client / Server Model** – An industry standard system design that allows multiple user applications (clients) to connect to a common service application (a server) to acquire common data types.
- **Integrated Development Environment** – A software development tool that provides a source code editor, source compiler, and additional development tools within a common application environment.
- **Graphical Editor** – An editor application that allows the user to modify software objects graphically (a visual of the object).
- **Windows Socket Based Communication (Winsock)**– A method of digital communication created by the Microsoft Corporation. This communications method is based loosely on an original communications concept developed at the University of California Berkeley that utilizes Internet Protocol (IP) to establish a data socket between two software processes. This socket can then be used to share data between the processes.
- **Voice Switching Control System (VSCS) –** This system is the fielded communications system for all EnRoute operations within the FAA
- **Active X –** A loosely defined set of technologies developed by Microsoft for sharing information among different applications. ActiveX is an outgrowth of two other Microsoft technologies called OLE (Object Linking and Embedding) and COM (Component Object Model).
- **Digital Signal Processing -** Refers to manipulating analog information, such as sound or photographs that has been converted into a digital form. DSP also implies the use of a data compression technique. A digital signal processor is a special type of processor designed for performing the mathematics involved in DSP. Most DSPs are programmable, which means that they can be used for manipulating different types of information, including sound, images, and video.

5 IATS VCS Introduction

During the fall of 2004, the Integration and Interoperability Facility (IIF) initiated a development effort that would provide communications services within the IIF Air Traffic laboratory with a user interface that is similar to the fielded En Route Voice Communication system called Voice Switch Control System (VSCS). The system was appropriately titled Simulated VSCS (SVSCS).

The SVSCS system is comprised primarily of Commercial off the Shelf (COTS) products.

The IATS program also had a specified need for an air traffic control communications system. The initial proposed solution from Lockheed Martin Air Traffic Management (LMATM) was to use a modified VTABS system. This system is used as a backup system in En Route operations and is very similar to VSCS. VTABS however, turned out to be very inflexible and cost prohibitive. In the process of finding suitable alternatives for voice communication, the IIF was tasked by the program office (AUA 200) to evaluate SVSCS as a solution.

After demonstrating the SVSCS to the program office and the Mike Monroney Aeronautical Center (MMAC) organization in December 2003, the EIIF was tasked with developing the IATS VCS as the communications system for the IATS program.

6 System Description

6.1 IATS High Level Description

The IATS is an En Route air traffic training system that is comprised of twenty-two smaller independent systems called training sectors. Each training sector is capable of providing training capabilities to support one En Route sector which includes a Radar position (R position) and a Data position (D position). In addition to the R and the D position each training sector includes two instructor positions (one for each trainee) and two ghost pilot positions. The ghost positions provide simulation capabilities for aircraft and adjacent ground entities. The twenty-two training sectors are grouped into the three laboratory configurations: Laboratory 1 (full lab), a Laboratory 2 (full lab), and a Mini lab. The full labs are comprised of ten training sectors each. The smaller mini lab contains the remaining two training sectors.

6.2 IATS VCS High Level Description

The IATS VCS System is comprised of four different subsystems. These subsystems are:

- VCS Communications (COM) Server Subsystem
- VCS Digital Signal Processor (DSP) Subsystem
- VCS Client Subsystem
- VCS Support Subsystem

Each VCS of the three VCS labs contains one COM Server subsystem. The Main laboratories contain 3 DSP subsystems and the Mini Laboratory contains 1 DSP. Each lab contains multiple clients (1 per training sector position). The COM Server, the DSP(s) and the multiple clients are connected through two different Ethernet Virtual Local Area Networks (VLANs) each configured on a Cisco Layer Two switching device. One VLAN connects the clients and the COM Server using Internet Protocol (IP), the second connects the DSP and the COM Server also with IP. Analog audio lines from the VCS DSP Audio Break-Out Boxes (BOBs) are run to each client, providing input and out signals for communication. The COM Server directs communication switching at the DSP based on requests received from client processors over the IP network. Clients also have analog signal lines run for detection of Push To Talk (PTT) which is enabled when a training sector position user intends to talk over that position's audio line. These analog lines all run to a PTT chassis that interfaces to the COM Server for evaluation. Clients may only communicate with clients in the same training sector. Figure 6-1 is a high level diagram of one of the VCS Main Labs.

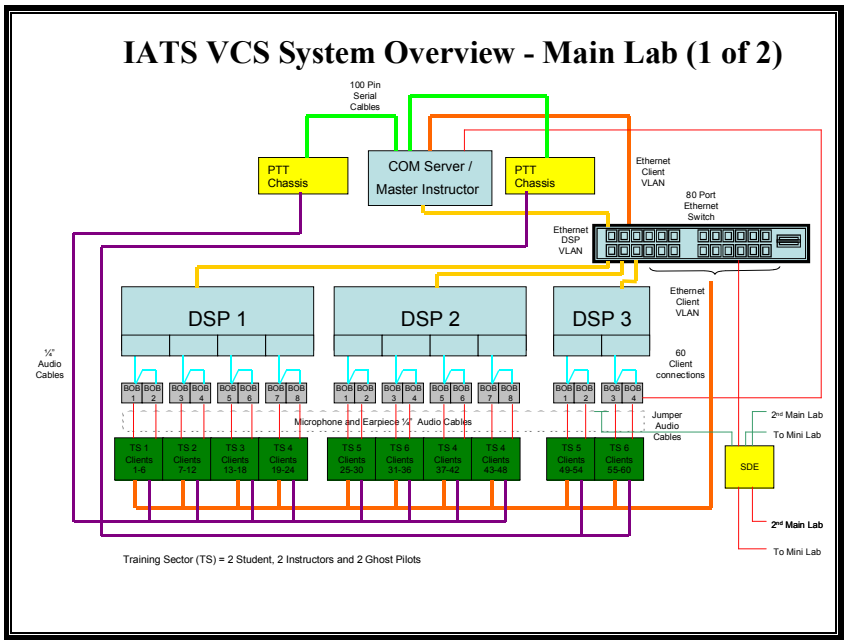


Figure 6-1 IATS High Level Design – Main Laboratory

The IATS VCS System for the mini lab will be essentially identical to the two main labs, the difference being the amount of training sector positions the system will support. The IATS VCS Software Development Environment (SDE) is integrated into the VCS Mini laboratory. Figure 6-2 is a high level diagram of the VCS Mini Labs.

Figure 6-2 IATS High Level Design – Mini Laboratory

6.3 VCS COM Server Description

The VCS COM Server is comprised of one processor running Microsoft Windows XP. As stated above, the server resides on two different virtual LANs, one to communicate with VCS clients including the SDE and the other to communicate with the DSP(s). This configuration ensures optimal server performance to both the DSP(s) and the clients. A 100 pin analog/digital acquisition card is installed in the processor and connected to the

PTT chassis. This card is used to detect PTT signals coming from client key devices (located in series with the headset). The key device is pushed when a user intends to talk over that position's audio line.

The COM Server allows client positions to connect to it and request communication connections to other clients within the same training sector. These requests are then disseminated to the DSP. The COM Server can also control startup and shutdown of all clients singularly, by training sector, or clients in the entire lab.

The VCS COM Server software was developed primarily using Visual Basic Version 6.0. An additional module for client control using Remote Procedure Call (RPC) was developed in Visual C++ Version 6.0.

6.4 VCS DSP Description

The VCS DSP is a complete hardware/software solution called SMx. SMx is manufactured by the Simphonics Corporation. SMx is comprised of one COTS server processor running Microsoft Windows XP and installed with the COTS SMx software application developed by Simphonics. The processor is also configured with interfaces to multiple Audio BOB devices. The BOB devices are used for analog audio routing between VCS client position audio channels.

The SMx application allows audio to be switched between audio channels based on logic and input data from the COM Server. The audio switching logic is developed using a Simphonics Development Environment and graphical language called V+. The graphical logic, which resembles flow chart diagrams, are contain within SMx proprietary files called Description Sheets. The SMx DSP application allows these Description Sheets to be loaded and executed.

6.5 VCS Client Description

The VCS Clients are the End User Interface (EUI) in the VCS system. This interface is comprised of a client processor running Microsoft Windows XP with a custom application that mimics the fielded VSCS system's EUI. The EUI is displayed on a touch screen LCD at each training sector position. The clients also contain audio channel inputs and outputs for voice communication as well as an input to the PTT BOB.

The VCS clients can be configured for one of three roles: student, ghost pilot, or instructor. The student client LCD and audio interface is integrated into ATC Display System Replacement (DSR) console, mimicking the En Route VSCS. The instructor LCD and audio is also integrated into the DSR console. The LCD is attached using an articulated arm and the audio headset is integrated using the supervisor preempt jack interface on the DSR console. The ghost pilot position clients are co-located with the IATS Signal pilot positions using a freestanding LCD screen identical to the student and

instructor interface and COTS audio interfaces. A detailed diagram that represents all the VCS components required for one VCS training sector is depicted in Figure 6-3.

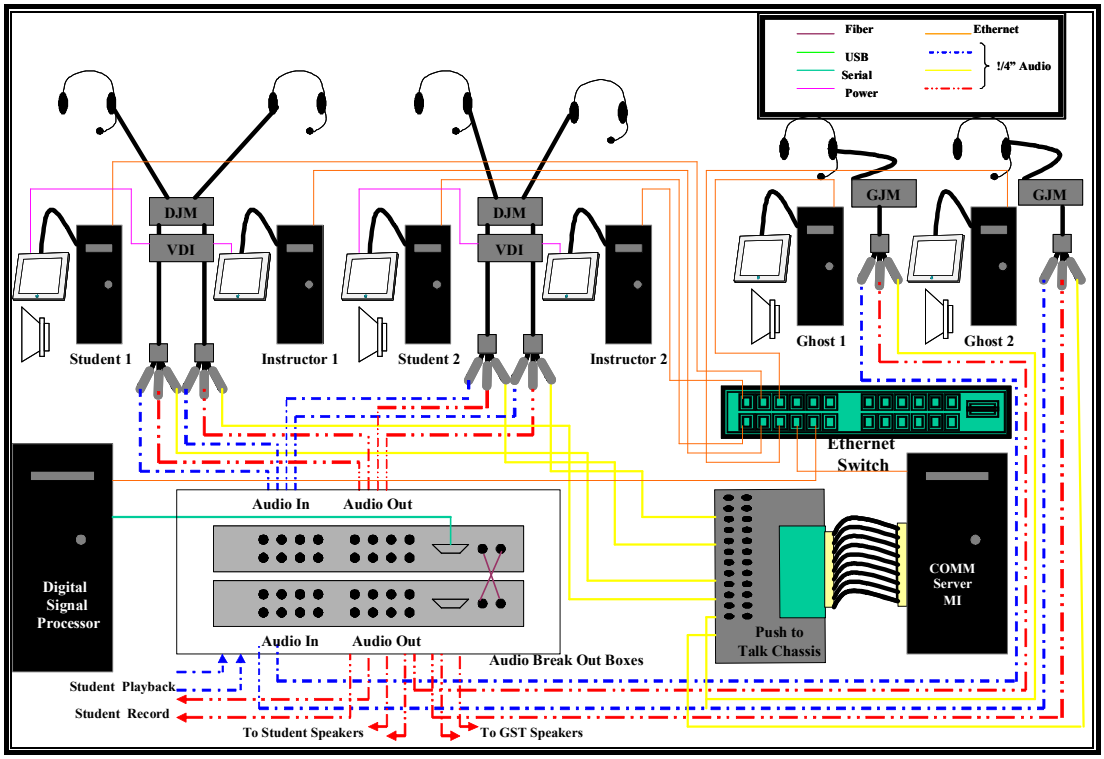


Figure 6-3 VCS Training Sector Architecture Detail

6.6 VCS Software Development Environment Description

The VCS SDE Processor resides in the Mini Laboratory and interfaces to the rest of the system in the same fashion as a VCS client. The processor contains the complete development environment software suite necessary for developing and maintaining all areas of the VCS system. This suite includes:

- Visual Studio 6.0 for Visual Basic and Visual C++ COM Server and VCS Client development
- SMX V+ for viewing and development of DSP description pages
- The IATS VCS Adaptation Builder for maintaining VCS adaptation

- All ActiveX development COTS licenses

The SDE design allows the developer to connect the SDE to the Laboratory 1, Laboratory 2, or the Mini Laboratory COM Servers as any adapted VCS client while executing thsoftware in the COTS development environment. It also allows the developer to view and modify the COM Server software and/or the DSP V+ Description sheets without interrupting Laboratory operations.

The SDE will also have a set of audio cables that can be plugged into the mini lab BOBs in place of any of the other clients. This will facilitate client debug with audio functionality using any client position.

The SDE is networked to all three laboratories on the Client VLANs.

7 VCS Hardware

7.1 Recommended VCS COM Server Processor Hardware

The COM Server was implemented in the IIF VCS test bed using a Dell workstation model PWS360. This model has a 3 Gigahertz (GHz) processor and is installed with 512 MB of RAM. The Server also had the following hardware:

- 19 Inch flat screen monitor
- National Instruments Data Acquisition PCI card. Model PCI-DIO-96
- Standard integrated keyboard, mouse, and VGA interfaces
- Two Fast Ethernet Network Interface Cards (NIC)s

7.2 Recommended VCS Client Processor Hardware

The VCS Client processors used in the IIF VCS test bed were Dell workstations model PWS340. This model has a 2.8 GHz processor and is installed with 256 MB of RAM. The clients also had the following hardware.

- GVision 10.4" LCD Touch Screen Monitor. Model JLPS-DW
- One Fast Ethernet NIC
- One Serial Communications Port (COM) port (required by touch screen)
- Standard integrated VGA interface

7.3 VCS DSP Hardware

7.3.1 Break Out Box Description

The BOBs connect to the DSP either directly as a master BOB or indirectly as a slave BOB via direct connection to a master BOB. Master BOBs use a standard serial cable connection to the SMX manufactured Peripheral Component Interconnect (PCI) card installed in the Server. Slave BOBs are connected to the master BOB via dual fiber connections. All Master BOBs connected to a DSP are also linked together via a sync cable that daisy chains them together. Figure 7-1 shows the proper DSP cabling configuration to support the VCS system.

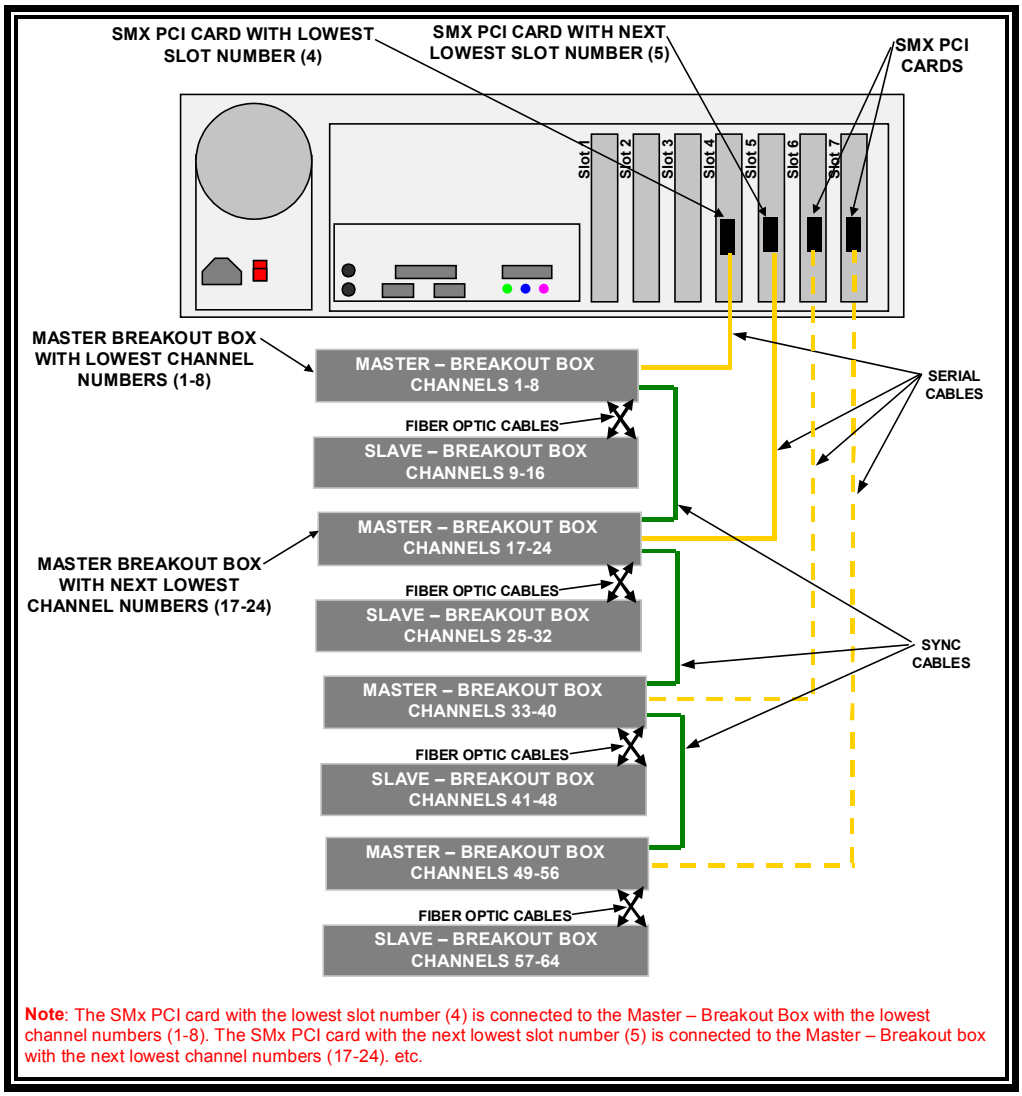


Figure 7-1 VCS DSP Break Out Box Configuration

7.3.2 DSP Break out Box Audio Channel Layout

Each IATS VCS training sector requires 15 audio channel pairs (input/output) to function. Each DSP BOB is comprised of 8 audio channels, thus two DSP BOBs are required per training sector – one master and one slave. A DSP at maximum capacity will accommodate 4 training sectors (8 BOBs). The audio lines within the BOB pair are allocated within the training sector as indicated in Figure 7-2. Note: BOB channel numbers for Training Sector 1 are shown. Subsequent TS channels are obtained by adding 16 to the previous TS within a DSP.

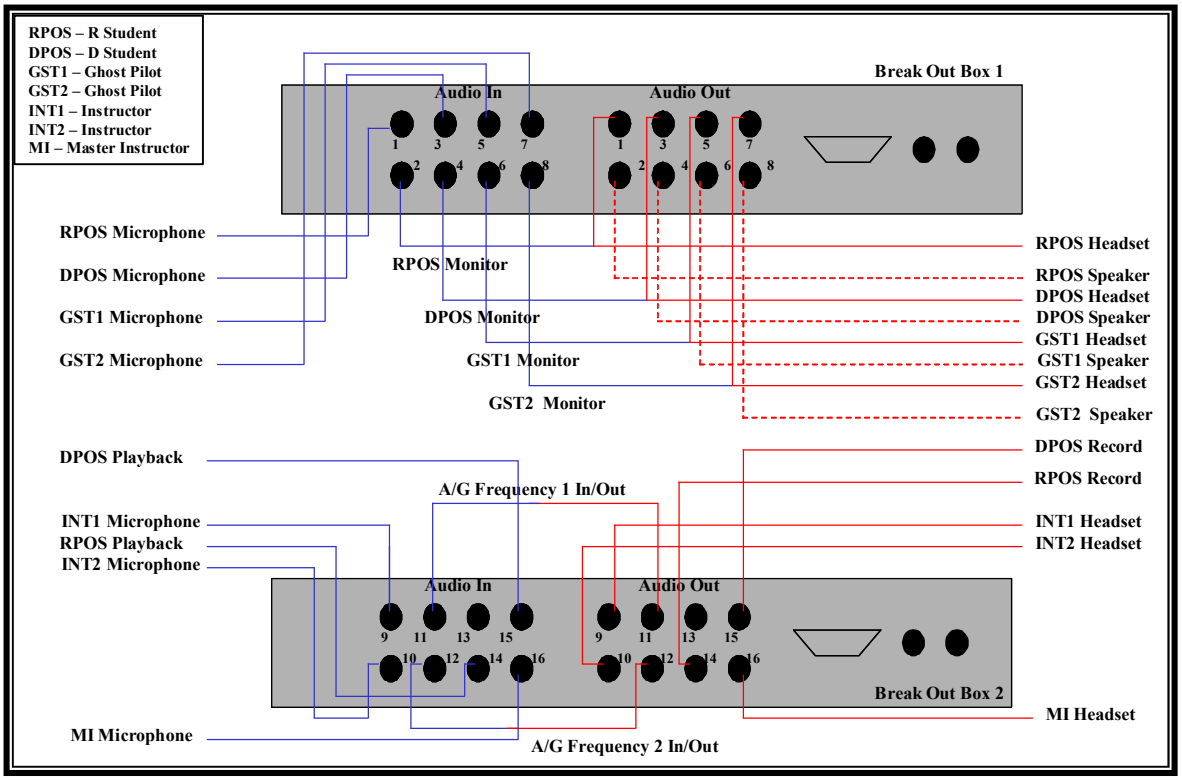


Figure 7-2 VCS DSP Break Out Box Audio Channel Layout

The student and ghost pilot positions are all allocated input channels for microphone, and output channels for headset and speaker audio. These positions are also allocated an input channel for position output monitoring. The headset output for each of these positions is fitted with an audio Y cable that directs the output in two directions. One lead is connected back into the BOB at that position's monitor input channel. The other

lead goes to the position's headset jack. The instructor positions are not required to have a monitor input channel or Y cabling.

There are two A/G frequency channel pairs allocated for each training sector (training sector Frequency 1 and Frequency 2). These pairs are "looped back" meaning the output interface for the channel is cabled back into the input interface of the channel. This configuration allows the VCS software to establish a sort of "party line" which is used to simulate A/G frequencies.

There are also input and output channel pair allocations for playback and recording of the student positions (two pairs) and a channel pair for Master Instructor communications to the training sector positions.

7.4 VCS Custom Hardware Description

7.4.1 VCS Hardware Parts List

The VCS system contains several custom hardware parts. These parts allow the VCS system to integrate with the Air Traffic DSR System and also simulate realistic PTT. There parts are listed as follows:

- The VCS Jack Module Adapter (JMA)
- The VCS to DSR Dual Jack Module (DDJM) Interface (VDI)
- VCS VGA cutout device
- VCS DSR speaker interface cable
- The VCS PTT chassis

7.4.2 VCS Jack Module Adapter

The VCS JMA is a VCS part common to all VCS positions. It is used for interfacing the Student DSR Headset Jack Modules through the DDJM or the Ghost Pilot Headsets through the Ghost Pilot position Jack Modules (GJM) to the VCS. It is comprised of a DB15 female connector that breaks out into three pairs of analog leads with three Male ¼" audio connectors. Two audio connectors are for Input and Output to/from the VCS DSP BOB. The third connector is for connectivity to the VCS PTT Chassis. Figure 7-3 depicts the VCS JMA with a DB15 pin out.

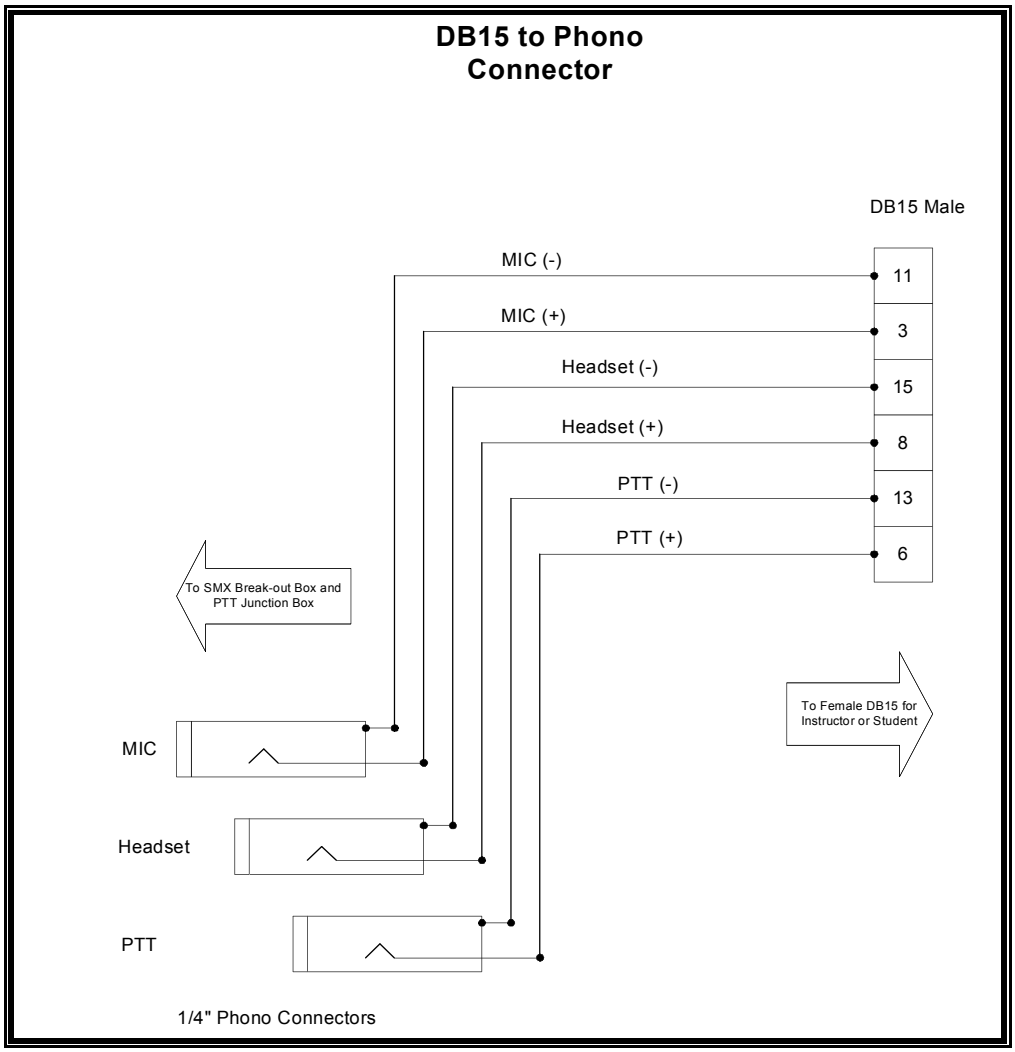


Figure 7-3 VCS Jack Module Adapter diagram with DB15 pin out

7.4.3 VCS – DDJM Interface

At the IATS student positions, the VCS is integrated into the Air Traffic automation portion of the system similarly to VSCS. The touch screen interface is mounted into the DSR console footprint and the VCS student and instructor audio interfaces use the DSR Dual Jack Modules. These jacks are designed to accept air traffic audio headset plugs.

Each side of the DSR console contains two jacks (1 DDJM). Of the four total audio jacks (2 DDJMs), the inboard jacks are wired as a pair, and the outboard jacks are wired as another pair. The VSCS system uses the inboard pair for the controller and the outboard pair as an Area Manager preempt jack. The VCS system uses the inboard pair for the student and the outboard pair for the instructor.

The VDI facilitates the interface from the DDJM within the DSR console to a pair of VCS JMAs (one for the student and one for instructor). It also provides two leads to the VGA cutout devices; one for the student and one for the instructor touch screens. The VDI cabling is housed within the DSR Console just below the VCS Touch Screen. Figure 7-4 is a diagram of the VDI custom cabling layout within the DSR Console.

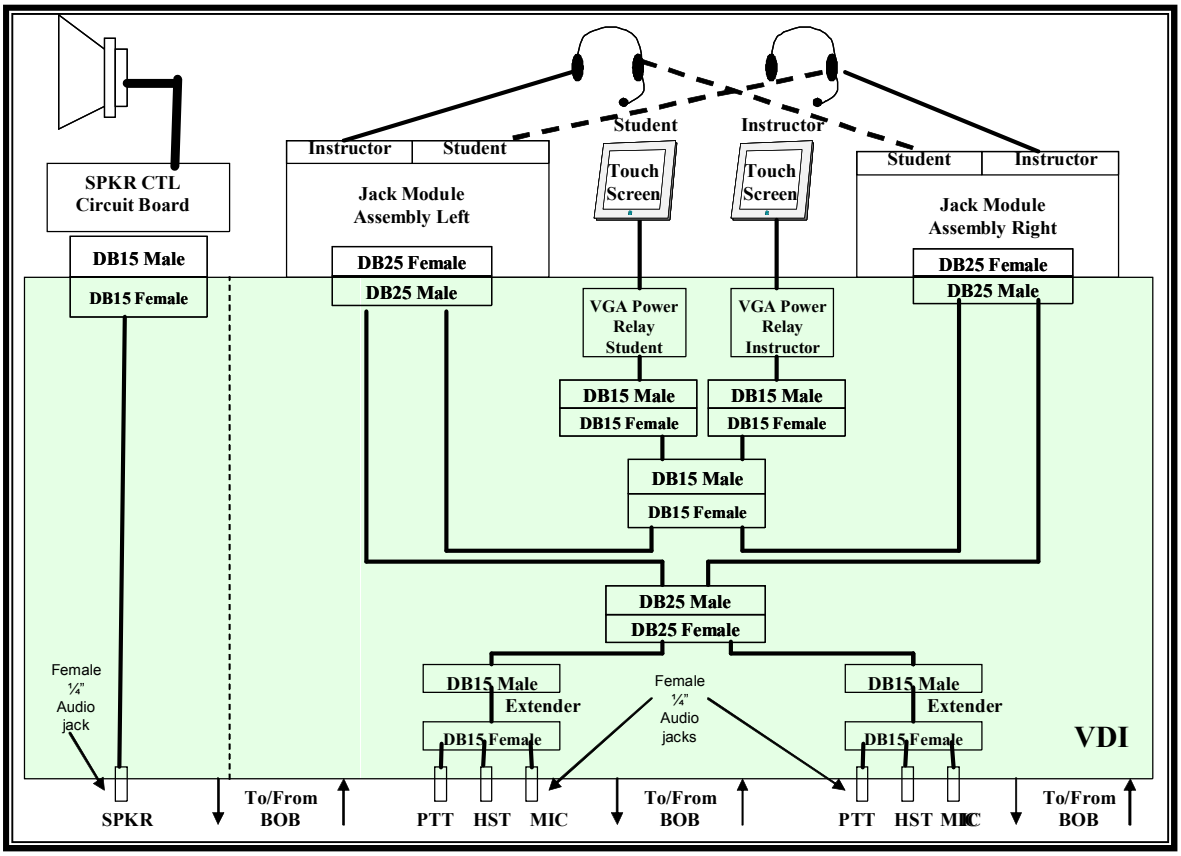


Figure 7-4 VCS – DSR DDJM Interface and VCS Speaker Cabling Layout within DSR Console

7.4.4 VCS VGA Cutout Device

The VCS VGA cutout device shuts down (removes power from) the VCS touch screen monitor for a position when that position's headset is removed from the jack. This keeps the screen from getting burn in and also represents the VSCS Display Monitor (VDM) function. The VGA cutout device is connected between the monitor power source and the monitor power interface. The VGA cutout device is controlled by a lead from the VDI. When a headset jack is plugged into the DDJM, it completes the VDI control lead circuit, which energizes the relay and provides power to the touch screen monitor. When the headset plug is pulled from the jack, the control circuit is opened and power is removed from the touch screen monitor. Figure 7-5 depicts the circuit diagram for the VGA cutout device.

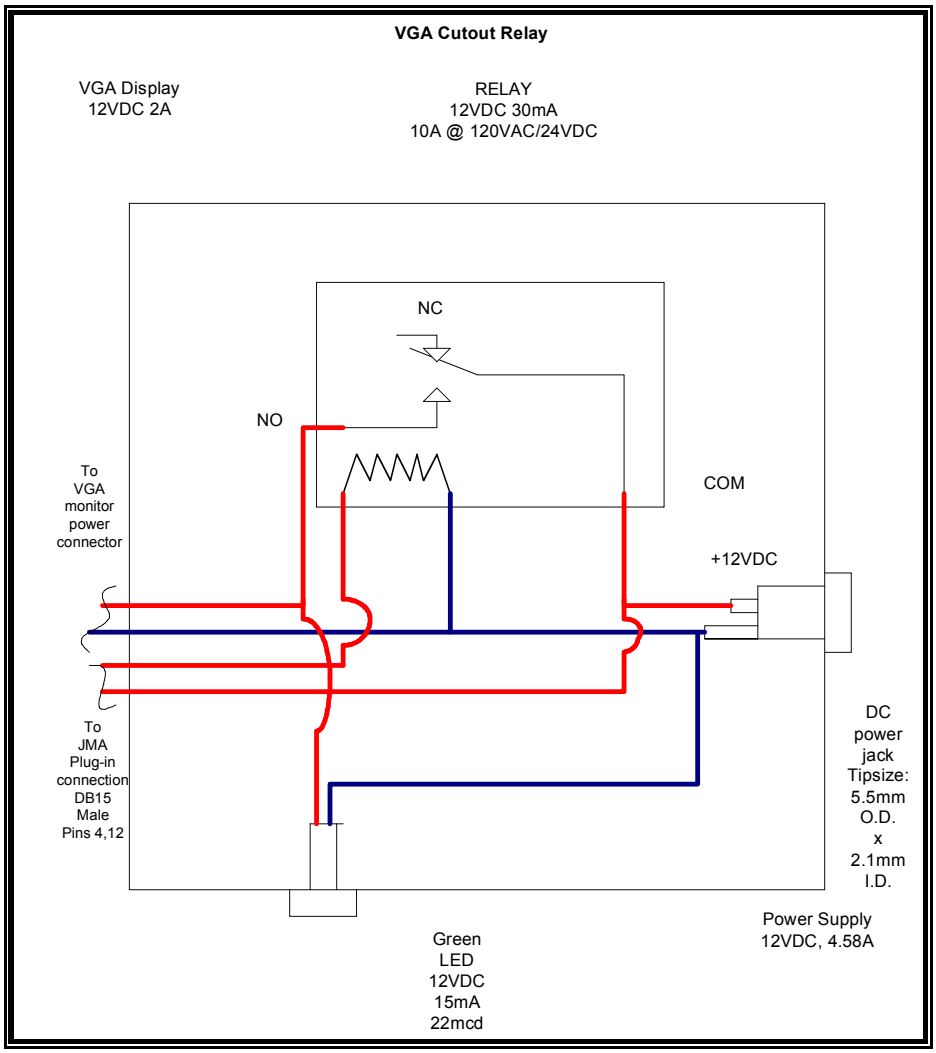


Figure 7-5 VCS VGA Cutout Device Circuit Diagram

7.4.5 VCS DSR Speaker Interface Description

The VCS system uses the A/G Audio speaker that is built into the DSR Console. A custom interface cable connects this speaker to the VCS BOB output in place of the interface cable that would connect the speaker to the VSCS. The cable has a DB15 female connector that connects to a DB15 male connector which is the DSR interface to the speaker. The other end of the cable in fit up with a ¼" audio connector. Figure 7-6 depicts the VCS DSR Speaker interface Cable.

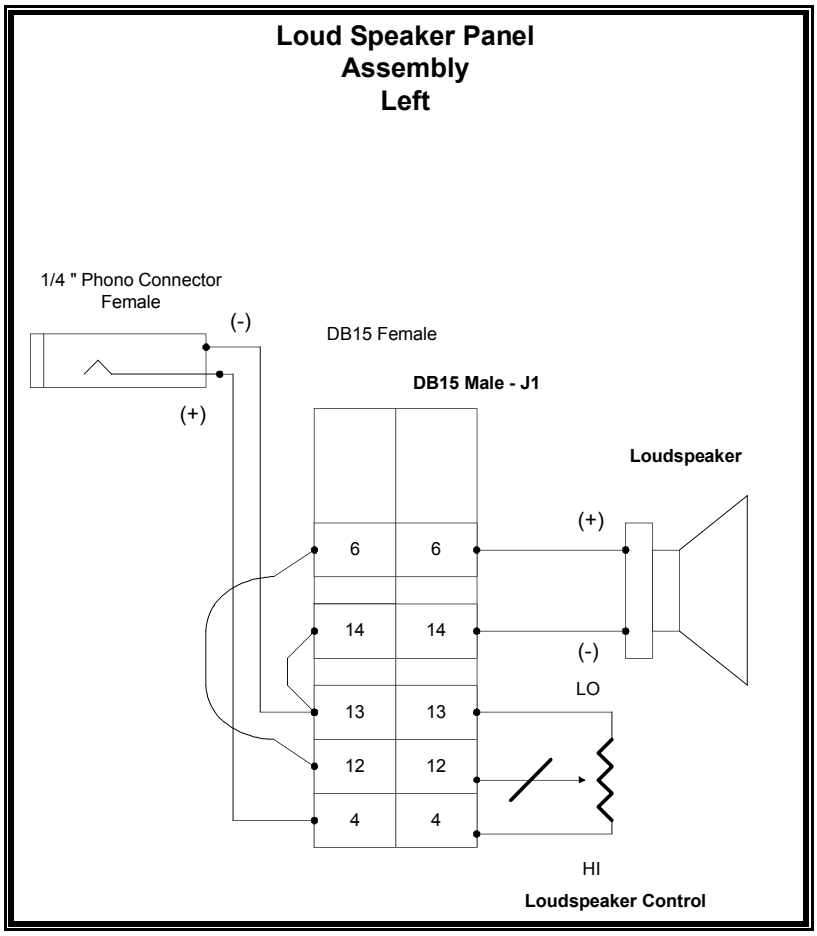


Figure 7-6 VCS DSR Speaker Interface Cable

7.4.6 VCS Push-To-Talk Chassis

The VCS PTT Chassis is a rack mounted custom hardware device that accepts ¼" male Audio inputs from VCS client positions. These inputs provide a signal to the PTT chassis when the user "keys up" on their audio headset to talk. These signals are forwarded to the VCS COM Server through a 50 pin breadboard that is mounted inside the chassis. The breadboard is connected via a 50 pin serial cable to the National Instruments Data Acquisition Card that is installed in the COM Server. The card actually contains a 100 pin connector that accommodates two 50 pin connections from two different breadboards thus allowing connections for two PTT chassis's per COM Server.

Each PTT chassis will accommodate up to 48 PTT inputs. Each main lab within the IATS VCS system has two PTT chassis installed allowing up to 96 inputs per lab (the main labs requires 60 inputs). The MINI lab requires only one PTT chassis using 16 ports out of the 48 available.

Figure 7-7 is a diagram depicting the PTT Chassis.

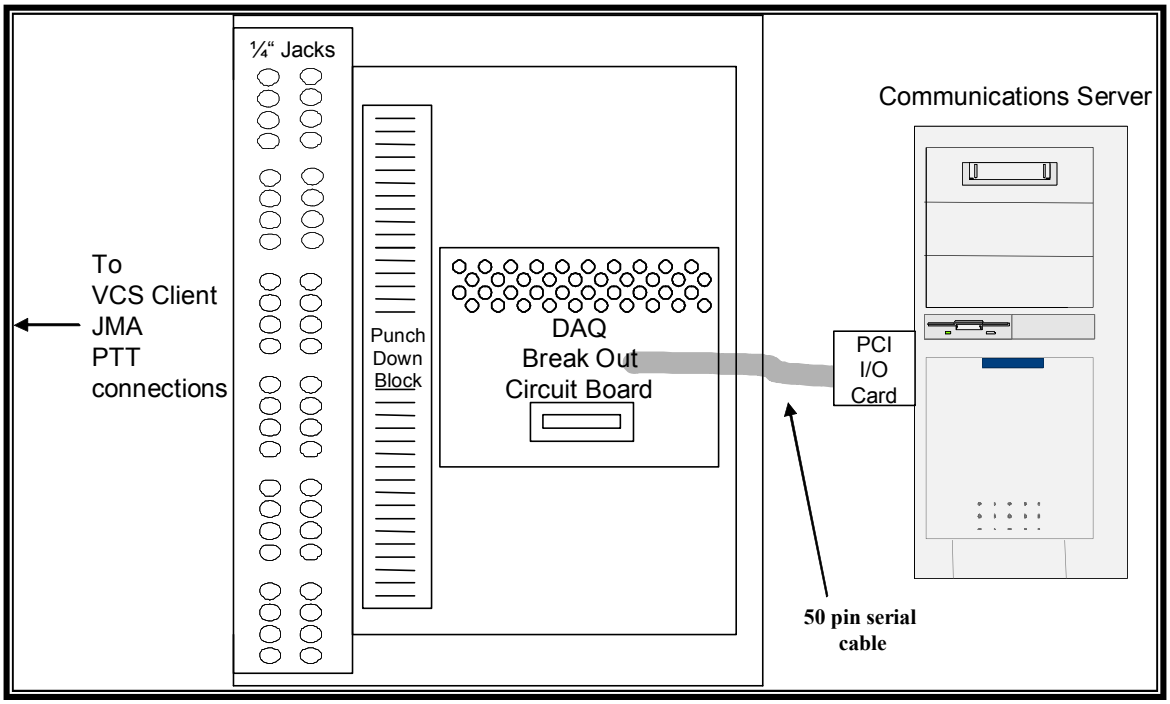


Figure 7-7 VCS Push To Talk Chassis

8 VCS Software Design

8.1 VCS COM Server Software Design

8.1.1 High Level Description

The IATS Server software is written in Visual Basic 6.0 using the Visual Studio 6.0 Integrated Development Environment (IDE). It provides a number of forms that a user can interact with to control certain functionality of the VCS system. When the user selects pull down menu choices or selects other objects in the main form, software event handlers process the action and perform the programmed response. All external GUI events occur asynchronously and are processed via a single threaded program. Additionally, Visual Basic program timers are used to ensure the timely operation of reading/writing to TCP and UDP socket ports as well as reading from the National Instruments (NI) Digital Input/Output (DIO) card to detect Push-To-Talk at client positions.

A simple message handler and parser are used to extract data from messages received from clients. The data in these messages may be used by the server to set (enable) the appropriate locations within a defined data buffer. This data buffer is then sent to the appropriate DSP when a program timer expires. The DSP processes the data buffer and enables or disables the audio circuitry that provides the requested client functionality. If necessary, client data messages are also used by the COM Server to contact a particular destination client with request information from the source client as in the case of an incoming call request. Figure 8-1 depicts the functional thread as described above from VCS client to DSP.

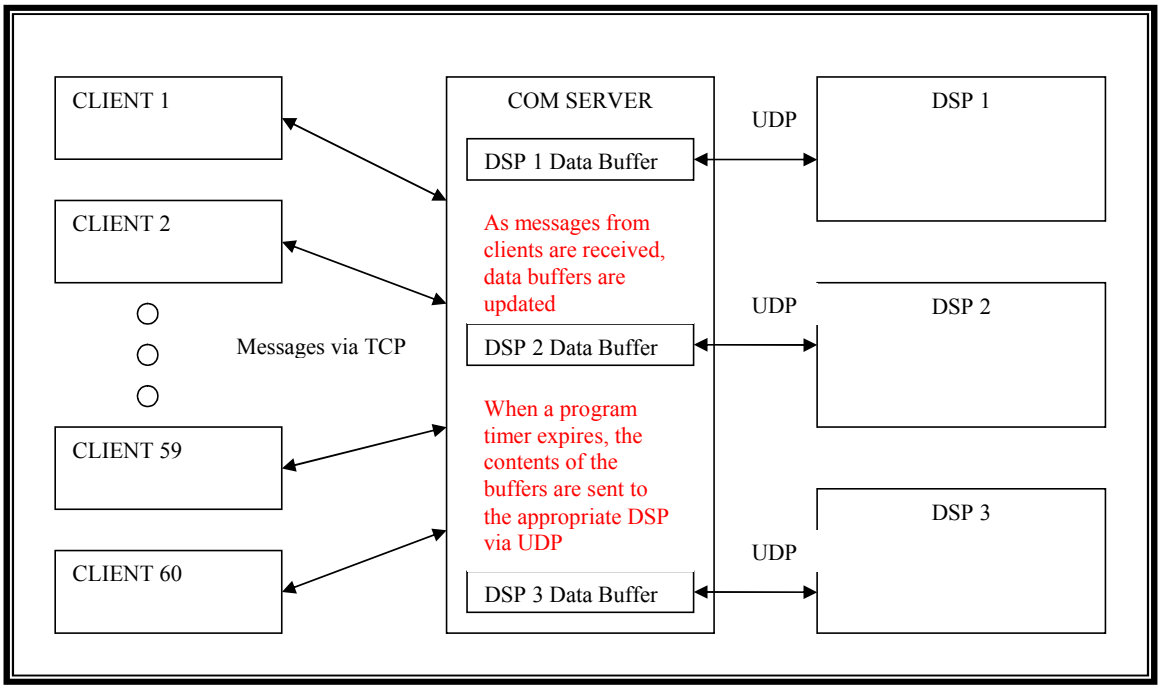


Figure 8-1 VCS High Level Software Communication Architecture

8.1.2 VCS COM Server Main Form Design Details

8.1.2.1 Main Form Details

The "Main" form is used to present connectivity status of the 60 external VCS clients, three external DSP systems, and the COM Server itself. Grey objects indicate non-adapted external systems. Yellow objects indicate adapted external systems. Green objects indicate adapted and connected VCS clients or DSP systems. Tool tip text is provided for most objects on this form when the mouse pointer is hovered over them. Tool tip text typically includes Hostname, Position identifier, IP address, or UDP Port. Additionally, the "Main" form provides menu-bar control functions to control client functionality and additional COM Server options.

The COM Server can start and/or stop client software in all Training Sectors, selected Training Sectors, or individual clients. It can also distribute new client software executables to these clients when necessary. Figure 8-2 depicts an active COM Server application and an adapted and connected DSP1. In this figure there are only 5 connected clients out of the 12 adapted as depicted on the main form. Volume controls are provided at the bottom of the form to allow the Master Instructor to adjust each Training Sector's tonal volume separately.

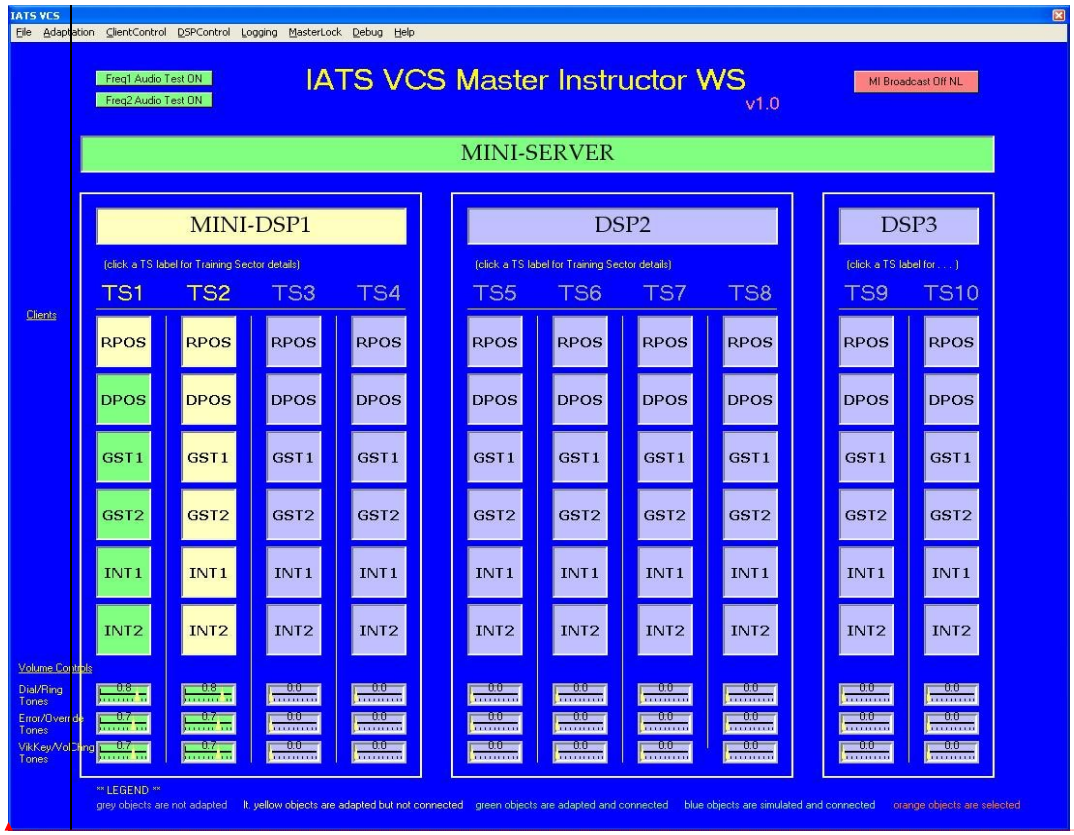


Figure 8-2 COM Server Main Dialog

8.1.2.2 Main Menu Details

The "Main" form main menu bar provides the Master Instructor with the capability to control training exercises by controlling the training exercise environment. Self-explanatory main menu bar capabilities include:

File->Exit

Adaptation->Load Adaptation

ClientControl->Software->Start All Training Sector Clients
ClientControl->Software->Start Selected Training Sector Clients
ClientControl->Software->Start Selected Client(s)
ClientControl->Software->Stop All Training Sector Clients
ClientControl->Software->Stop Selected Training Sector Clients
ClientControl->Software->Stop Selected Client(s)

ClientControl->Software->Distribute Latest Client Software to All Clients
ClientControl->Software->Distribute Latest Client Software to Selected Clients
ClientControl->Software->Restore Previous Client Software Version to All Clients
ClientControl->Software->Restore Previous Client Software Version to Selected Clients
ClientControl->Hardware->Reboot All Clients
ClientControl->Hardware->Reboot Selected Training Sector Clients
ClientControl->Hardware->Reboot Selected Client(s)

DSPControl->Software->Start All DSPs
DSPControl->Software->Start Selected DSP
DSPControl->Software->Stop All DSPs
DSPControl->Software->Stop Selected DSP
DSPControl->Hardware->Reboot All DSPs
DSPControl->Hardware->Reboot Selected DSPs

Logging->Capture Minimum (default)
Logging->Capture Maximum
Logging->Viewer

MasterLock->Lock Form
MasterLock->UnLock Form

Debug->Heartbeat to Client (default)
Debug->Debug Heartbeat to Client (when running the COM Server in Debug Mode)
Debug->SDE Simulated Position (See Section 12)

Help->About

The ClientControl pull down menu displays are depicted in Figure 8-3a and 8-3b. The ClientControl->Software pull down (Figure 8-3a) allows the MI to start and stop Client software on adapted Client PCs. Options include all Client PCs, selected Client PCs, or all of the Client PCs in an entire training sector. The MI is also able to distribute the latest or a previous version of Client software to all of the Client PCs in an entire training sector or the entire lab. The ClientControl->Hardware pull down (Figure 8-3b) allows rebooting of individual Client PCs, all of the Client PCs in a single training sector or the entire lab.

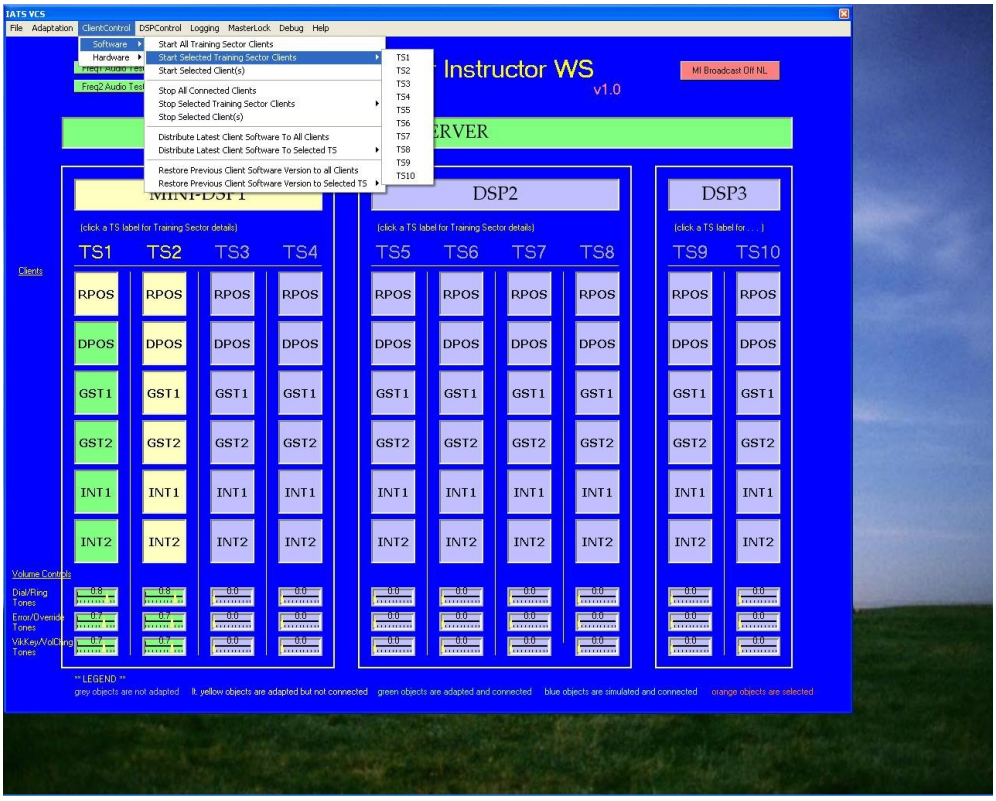


Figure 8-3a COM Server Main Dialog – Client Control Software Menu

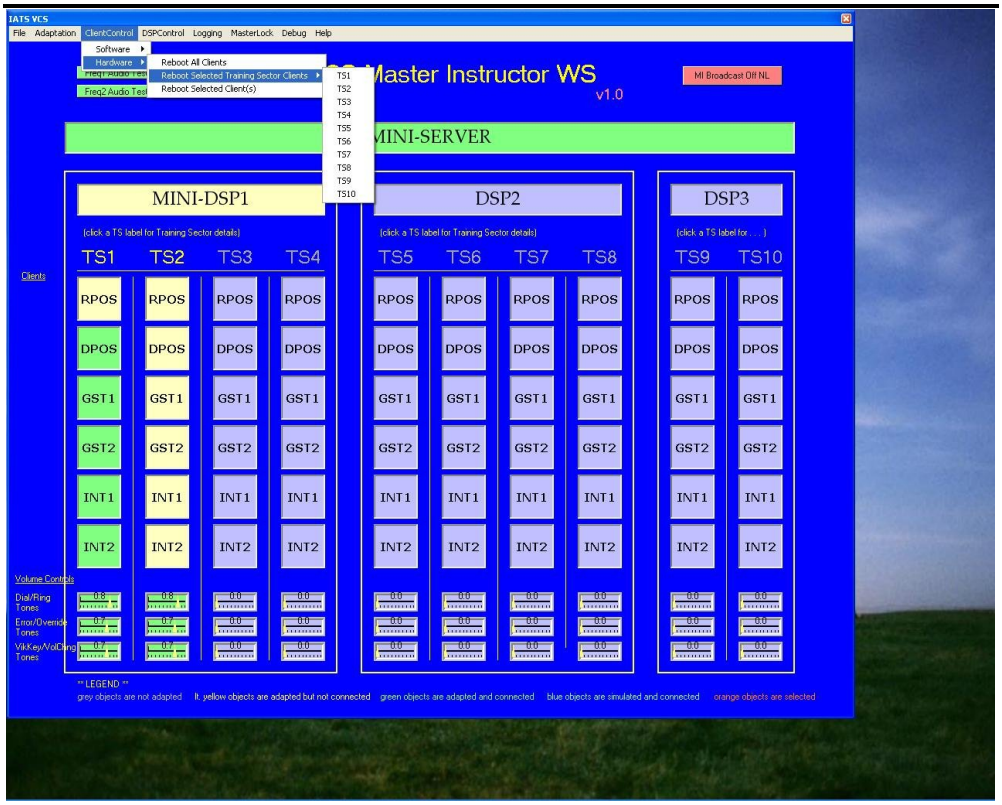


Figure 8-3b COM Server Main Dialog – Client Control Hardware Menu

8.1.2.3 Training Sector Audio Routing Grid Form Details

The "Training Sector Audio Routing Grid" form, depicted in Figure 8-4a, provides more detailed Training Sector information and is displayed by selecting (clicking) a training sector label on the Main form. This set of forms, one for each training sector, presents underlying communication buffer status by providing the current value of most of the DSP data buffer that is sent to each DSP every 70 ms. This data buffer contains the status of 2049 DSP Port objects, each represented by 32 bits within the buffer. Each DSP Port object is used by the DSP's automated analog-to-digital-to-analog multiplexing software. These port objects may also comprise discrete volume and gain control objects for each of the 16 input and 16 output channels allocated to each of the 10 adaptable training sectors. When a particular port indicator within the audio routing grid is colored green, it indicates that a specific 32 bit section of the buffer is "set", thus enabling a particular virtual thread and therefore a function, at the DSP.

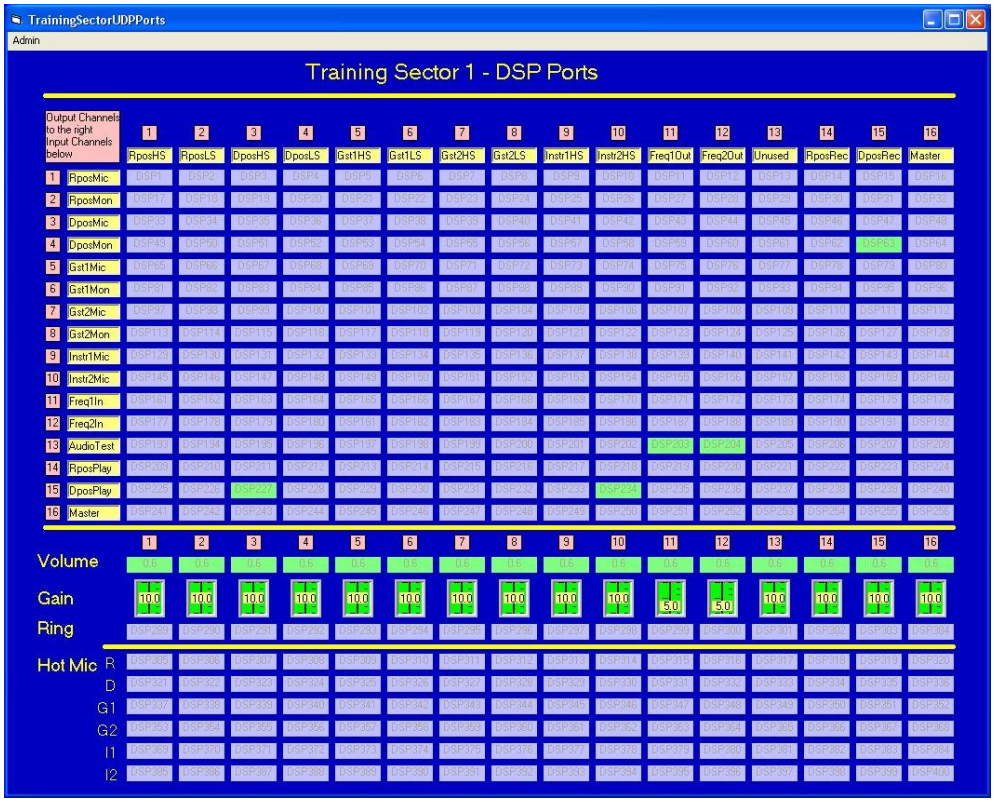


Figure 8-4a COM Server Audio Routing Grid Dialog - Locked

The “Training Sector Audio Routing Grid” form depicted in Figure 8-4b provides a method for changing UDP values within a Training Sector. The controls on this form can be locked by selecting the Admin pull down menu in the main menu bar of this form and selecting Lock. The controls on this form can be unlocked by selecting the Admin pull down menu in the main menu bar of this form and selecting UNLock, however the Admin password is required for this. If the form is unlocked, the UDP ports are available for manual enable/disable by clicking on the desired port object. If the form is locked, the UDP ports are NOT available for manual enable/disable by clicking on the desired port object.

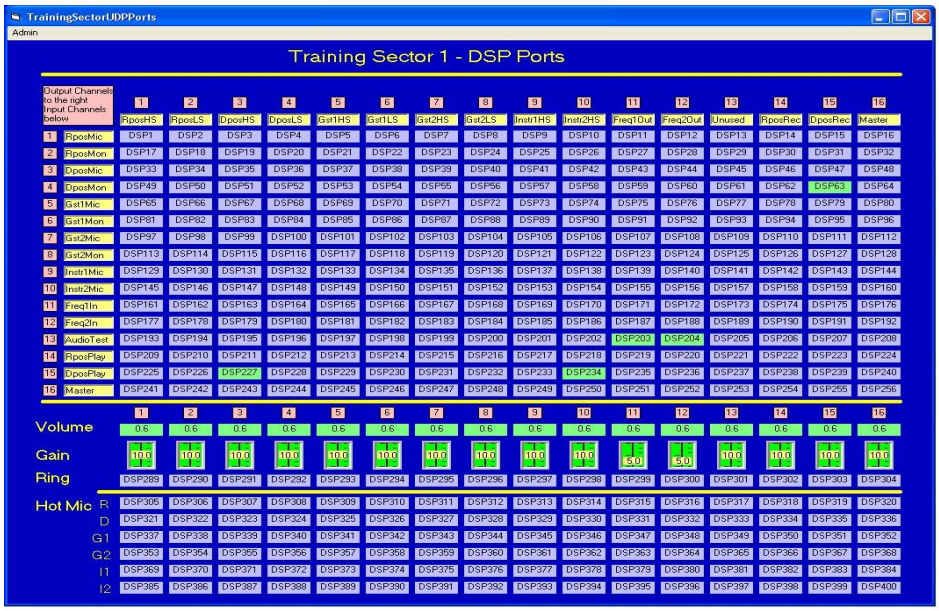


Figure 8-4b COM Server Audio Routing Grid Dialog - Unlocked

8.1.2.4 Log Viewer Form Details

A “Log Viewer” form, depicted in Figure 8-6 is used to convey meaningful date and time stamped status information pertaining to the messages transmitted and received between the clients and the server. This form is displayed via the “Main” form’s main menu-bar “Logging” selection and then selecting “Viewer” (see Figure 8-5).

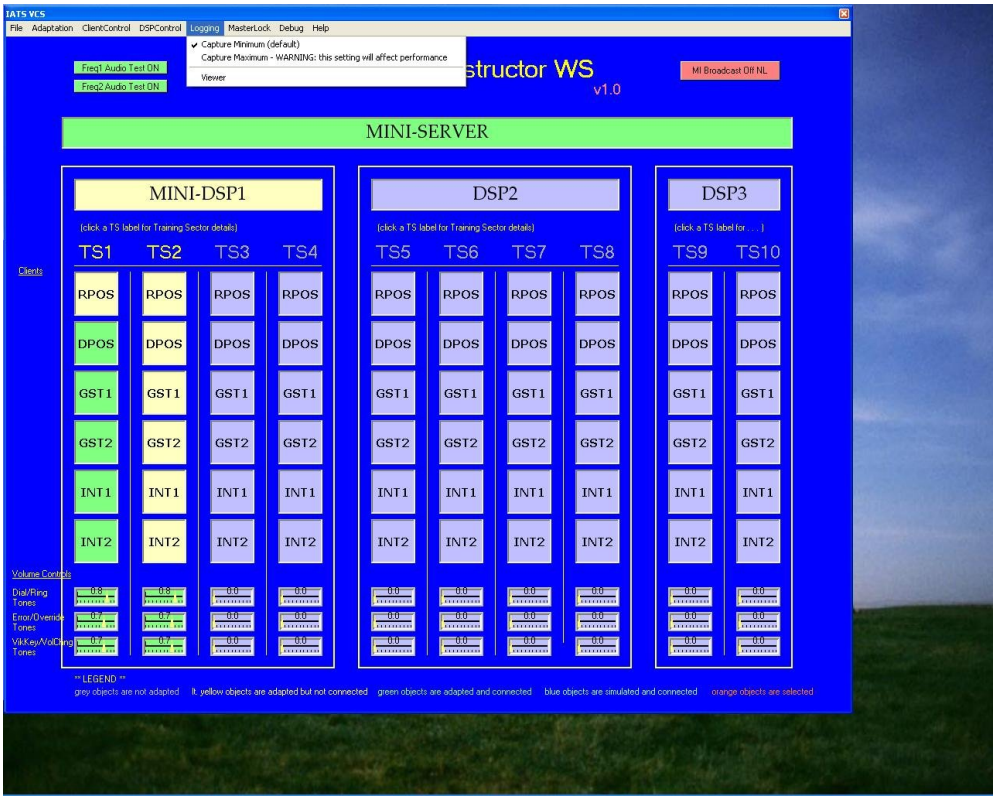


Figure 8-5 COM Server Main Dialog – Logging Menu

The “Log Viewer” form itself is displayed on top of the main form and can be disposed of by clicking on the “x” in the upper right hand corner of the log viewer form. Certain types of messages deemed to be the most significant are displayed when the logger is capturing log data in ‘Capture Minimum’ mode, which is the default mode (see Figure 8.5). In ‘Capture Maximum’ mode, the logger will capture and display much more data.

NOTE - This may affect the performance of the VCS system. Selecting ‘Capture Maximum’ mode is NOT recommended unless there is an absolute reason for doing so.

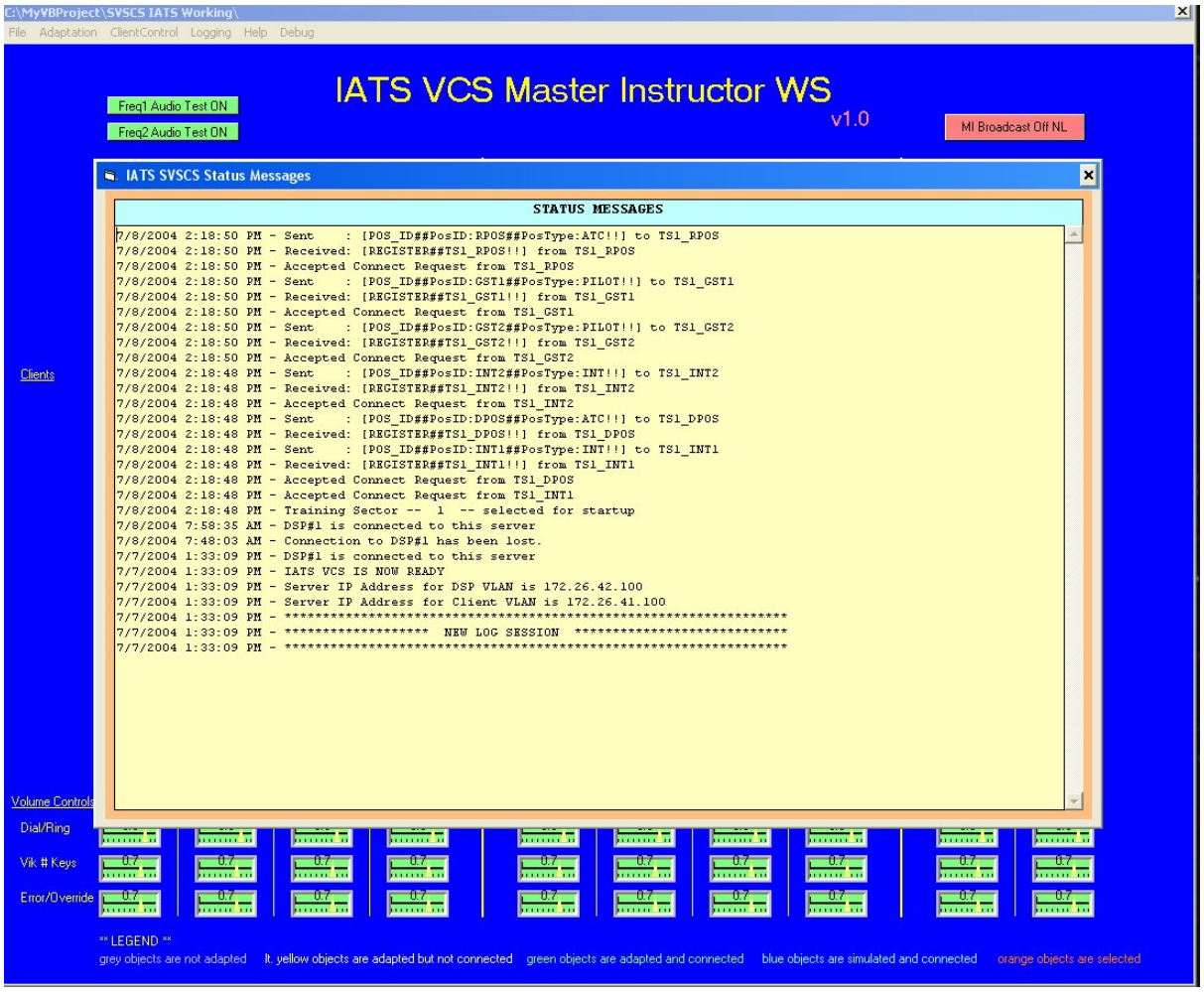


Figure 8-6 COM Server Status Message Dialog

8.1.2.5 Load Adaptation Set/Select Adaptation Set Forms Details

The "Load Adaptation Set" form and "Pick Adaptation Set" form, depicted in Figure 8-7, provide functionality to load a certain adaptation set into a specified training sector. All training sectors may be loaded individually with the same adaptation set or with different adaptation sets. An adaptation set defines the position identifiers for each of the six clients in a training sector as well as the Air-to-Ground (A/G) and Ground-to-Ground (G/G) screen formats displayed at these client positions.

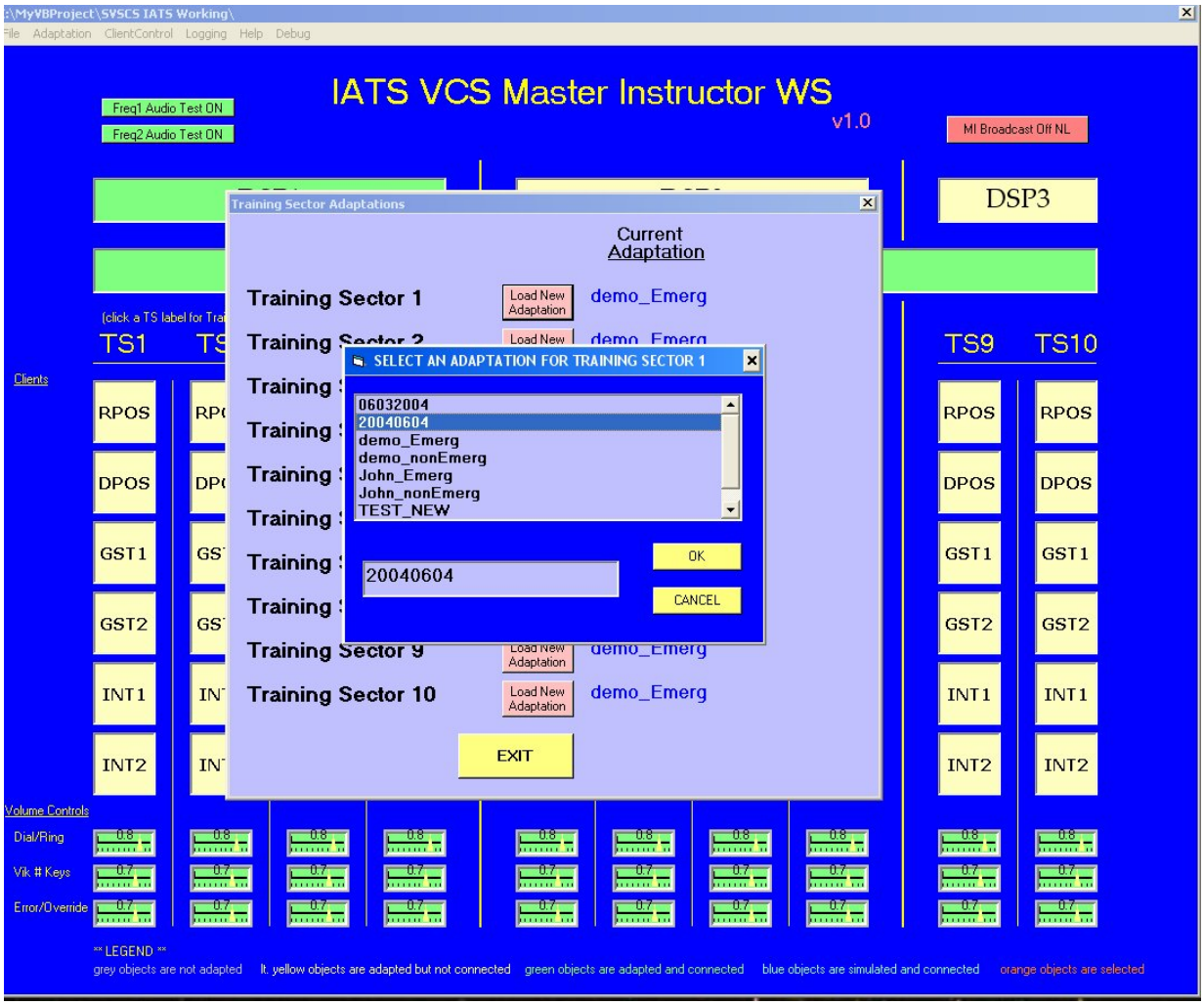


Figure 8-7 COM Server Adaptation Dialog

8.1.3 VCS COM Server Initialization Thread Overview

8.1.3.1 FormLoad (frmMain.frm)

The FormLoad( ) function in the frmMain.frm Visual Basic object module is analogous to a "main" function in other high-level programming languages and represents the only entry point of the executable. Prior to loading the "main" form (the FormLoad() functions's primary responsibility), this function initializes global variables and calls other functions such as, creating the COM Server LogFile, reading the HOSTS file and storing relevant information into internal data structures, creating and initializing the TCP and UDP sockets to provide communication paths to the clients and DSPs, and loading ancillary forms into memory for subsequent "showing".

8.1.3.2 clientStartAll, clientStartSelected, clientStartSlectedTS (frmMain.frm)clientStopAll, clientStopSelected, clientStopSlectedTS (frmMain.frm FormLoad (frmMain.frm)

The clientStart/Stop( ) functions are called when the Master Instructor starts/stops client applications remotely from the COM Server Application. The Master Instructor can remotely start or stop one, many, or all clients via a main menu bar pull down option. When one of these pull down menu options are selected, one of the clientStart/Stop functions is called. Each of these functions calls out to an external Remote Procedure Call (RPC) executable located in the same running folder as the COM Server application. The RPC executable acts as an RPC client application that communicates with an RPC server application running on each of the VCS client nodes. This method is used to remotely start/stop client application software on any client node.

8.1.3.3 Winsock1_ConnectionRequest (frmMain.frm)

Once a client is remotely started, it will attempt to connect to a well-known COM Server port. The incoming Connection Request will trigger the Winsock1_ConnectionRequest ( ) event handler. The remote HostName of the client attempting to connect to the COM Server is determined and validated in this function. If valid (i.e. client is represented in the HOSTS file), the connection request is "accepted" and certain connection information is stored in local data structures for subsequent program use.

8.1.3.4 Winsock1_DataArrival (frmMain.frm)

Once a client is connected to the COM Server, the COM Server expects to receive messages from it. Messages received from clients over their respective TCP/IP connections are received at the COM Server by the Winsock1_DataArrival( ) event handler. This event handler simply passes the received data to the clientParseData( ) function. The data is tokenized in the clientParseData( ) function before being passed on to the clientParseMessage( ). The clientParseMessage ( ) function evaluates the tokens. Message type is determined in the clientParseMessage( ) function by examining the first token. Using the message type, the clientParseMessage ( ) function calls the specific message handler function for that message type.

8.1.3.5 clientParseMessage (frmMain.frm)

The clientParseMessage function is a switch (case) statement that calls the specific message processing function based on the type of message received. Individual message types are described in more detail in Section 8.6 of this document.

Valid Air/Ground (A/G) message types include:

- AG_AUDIO
- AG_PTT
- AG_RT
- AG_SET_STATE

Valid Ground/Ground (G/G) message types include:

- GG_ANSWER_CALL
- GG_ORIGINATE_CALL
- GG_RELEASE_CALL
- GG_JOIN
- GG_LEAVE
- GG_VOICE_MONITOR

Valid other message types include:

- ERROR
- KEY_PRESSED
- PSN_REL
- REGISTER
- SIM_CLIENT_INFO_REQ
- SIM_DSP_INFO_REQ
- SIM_REGISTER
- VOLUME_CHANGE_HS
- VOLUME_CHANGE_LS

8.1.3.6 Process_REGISTER_Message (frmMain.frm)

Receipt of a REGISTER message from a client indicates that they are successfully connected and desire a copy of the adaptation data that resides on the COM Server. To accommodate this request, the COM Server locates the appropriate adaptation set using the HostName provided in the REGISTER message and pushes the data out to the client

machine using a series of TCP messages (see Section 8.6.3.1). Upon completion of this process, the client will have all the information required to function fully within the training sector.

8.2 VCS Client Software Design

8.2.1 High Level Description

The VCS Client software is written in C++ using Microsoft's Visual Studio 6.0 development environment. The Client application is comprised of seven GUI dialogue screens that facilitate user communication requests and react with the appropriate visual and auditory responses.

- Air-to-Ground 1
- Air-to-Ground 2
- Ground-to-Ground 1
- Ground-to-Ground 2
- Air-to-Ground Status
- Utility
- VEM Indirect Access Keyboard (VIK)

The VCS dialogue screens are constructed primarily of COTS ActiveX controls. These A/G and G/G screens objects are dynamically configured by adaptation data which is received from the COM Server at Client initialization/connection. The adaptation allows the application to present multiple user-selectable buttons to the user each representing various VCS communication types. These types include A/G and G/G calls.

Upon initialization, the VCS client retrieves its Computer Name (environment variable) and sends a TCP connection request to the VCS COM server, registering itself as a VCS client. When the TCP connection request is accepted, the COM Server sends the client the proper adaptation based on the client's Computer Name.

When a VCS user presses an A/G or G/G client call button, a request is sent via a TCP message to the VCS server. The client software presents the status of the request with a visual indication. Upon receipt of the request, the COM server directs the DSP to set up the appropriate communication paths and/or forwards the request to a specified destination client. The VCS client also updates the appropriate user screen as incoming messages are received from the VCS COM Server.

8.2.2 Design Details

The following 7 object classes are described in detail in this section:

- AgScreen.cpp

- MessageHandler.cpp
- AgScreenDlg.cpp
- GgScreen.cpp
- AgStat.cpp
- UtilScreen.cpp
- VIkScreen.cpp

8.2.2.1 VCS Client Application (AgScreen.cpp)

The only VCS Client application entry point is resides in the AgScreen Class which is represented by the AgScreen.cpp source code file. The CAgScreenApp object class is the main windows application class object. The main functioning method within this class is InitInstance(). Upon initialization, the InitInstance() method retrieves the VCS Client computer's hostname environment variable, instantiates the MessageHandler object and sets up the Message Handler's TCP socket as the application's communication thread for all client-server communication. When a TCP socket is connected, the object then constructs and associates the dialogue class objects preparing the client application for receipt of adaptation data from the COM Server.

8.2.2.2 Message Handler (MessageHandler.cpp)

The MessageHandler object class methods send, receive, process and log all client-server communications. A log file located in the C:\Executable\Log directory is created prior to any client-server communication. It is named <Hostname>_<Date_Stamp>.log (eg. Lab1-TS1-RPOS_071404.log). Error messages and user actions (i.e. button pushes) are also logged. All client-server message types are explained in detail in section 8.4 titled VCS Client/VCS COM Server Interface Control Documentation. Listed here are some common processing methods from the message handler class.

Common MessageHandler Argument Variables:

SrcId - String (CString type) variable that represents the ID of the client position (i.e. R66).

TrunkId - G/G call string variable that represents the trunk associated with a DA button.

CallId - G/G call string that represents the destination Id of a request.

Type - G/G call string variable representing a call type (eg. IP, IC, Override).

HollerOn - Boolean variable indicates that an IP call is of the type Holler or not.

Latching - Boolean representing a button characteristic. False indicates that the button will disengage when released

Freq - Air-to-Ground (A/G) frequency Id.

Site - Site Id associated with a particular A/G Frequency button.

Channel - Represents the channel Id associated with a particular A/G frequency.

8.2.2.2.1 Key MessageHandler Methods

AnswerCall(CString CallId, CString SrcId, CallType Type, CString TrunkId, bool HollerOn) This method sends a message to the server indicating that the client user is answering a ringing call.

CallJoin(CString SrcId, CString TrunkId) - CallJoin sends a message to the server requesting that the client be added to the previously established trunk when the client user presses an in use IP type DA (Direct Access) button. SrcId represents the client's ID (i.e. R66) and the TrunkId is the identifier of the trunk associated with the DA button pushed.

CallLeave(CString SrcId, CString TrunkId) CallLeave requests that the server remove the client from the trunk of an active IP call.

EmergencyPtt(CString Type, CString Action) Sends a message to the server indicating the client user's intent to talk on the emergency and non-emergency channels. Also known as Software PTT this method is associated with Air-to-Ground Emergency Frequencies. The Action parameter may be "ON" or "OFF".

Get_Token(CString Message, CString& Token, int startPos) This method captures a token from a string message beginning at a given starting point (startPos) and ending with a preset message token separator.

LogIt(CString LogMessage, LogItType MsgType) - The LogIt method expects two arguments. The first, LogMessage, is the actual string to be appended to the log file. The second, LogItType, identifies the message as either a sent, received, initialization or error message, or as a user action. The message type along with a date and time stamp are pre-pended to the log message before it is added to the log file.

MessageHandler() - This is the main message handler method that creates the log file, sets up the TCP socket for server communication and registers the VCS Client with the server.

~MessageHandler() - Upon exiting the message handler, this method closes the log file and disconnects the TCP socket.

OnClose() - When the VCS client's connected TCP socket is disconnected, a message box is displayed to the user indicating that the server has died and the VCS client application is exiting.

OnReceive(CString FullMessage) - The OnReceive method processes all incoming communications. The FullMessage argument is parsed and processed according to the type of message received. If a partial message is received it is buffered and pre-pended to the next incoming message.

OriginateCall(CString IdName, CString SrcId, CString CallType, CString TrunkId) This method sends a message to the server to initiate a ground to ground call. IdName in this function represents the callee or the destination of the intended call.

Parse_AG_Button_Data(CString Btn_Input, int Screen_Num) This method parses a A/G frequency button token string message from the server (Btn_Input) , builds the frequency button in the appropriate A/G screen and returns TRUE if the button was successfully built in the specified screen and FALSE if the data was not sufficient to build the button.

Parse_GG_Button_Data(CString Btn_Input, int Screen_Num) This method parses a DA button token string message from the server (Btn_Input) , builds the DA button in the appropriate G/G screen and returns TRUE if the button was successfully built in the specified screen and FALSE if the data was not sufficient to build the button.

QueueFull(CString CallId) - The QueueFull method sends a message to the server indicating that the Call Answer (CA) queue is full. The CallId argument identifies the ID of the calling client whose call can not be accepted.

ReleaseCall(CString CallId, CString SrcId, CallType Type, bool Latching, bool HollerOn, CString TrunkId) Used for all G/G call types, this method requests that the server deconstruct an active call.

SendSrvError(CString SrcId, CString DestId, CString TrunkId, CString ErrorText) SendSrvError method submits an error message to the server to be displayed at another client's message text box.

SendVikKeyToneMsg(CString Key, CString State) This method sends a message to the server triggering a key tone state "ON" when a VIK number key is pressed and canceling the tone "OFF" when the VIK key is released.

SetAssociations(CAgScreenDlg* pAG, GgScreen* pGG, UtilityScreen* pUTL, AgStatus* pAGS) SetAssociations is a scoping method which creates pointer variables to all other dialogue screen objects. This gives the message handler visibility to these dialogue screens.

SetState(CString Freq, CString Site, CString Channel, CString Function, CString Condition) The SetState method sends a message to the server setting the state of a particular Air-to-Ground frequency. The Function argument represents either Transmit (XMT) or Receive (RCV) and the Condition argument represents the intended state "ON" or "OFF".

SwitchAgAudio(CString State, CString Freq_Channel) This method sends a message to the server to switch A/G incoming audio routing for a particular frequency. The State argument in this method actually represents the headset (HS) or loud speaker (LS) and the Freq_Channel represents the channel to be routed.

SwitchAgRt(CString State) The SwitchAgRt method sends a message to the server to route the client's incoming Air-to-Ground audio to the speakers. The State argument is ether "ON" or "OFF".

VoiceMon(CString CallId, CString SrcId) The VoiceMon method sends a request to the server requesting a monitor of the position identified in the CallId argument.

VolumeChange(CString Output, int UpDown) VolumeChange sends a directive to the server to change the volume at a client position. The Output argument represents either the headset (HS) or loudspeaker (LS) and UpDown is indicated by an integer that represents the intended adjustment, "UP" (1) or "DOWN"(0).

8.2.2.3 Air to Ground Dialogue Screens (AgScreenDlg.cpp)

The Air to Ground Dialogue Screen allows the VCS client users to participate in simulated air to ground communications. There are two audio input/output channels dedicated to Air to Ground frequency communications. The VCS adaptation allows the two audio channels to be adapted as Emergency or Non-Emergency frequencies. Each A/G frequency button is mapped to one of the two audio channels.

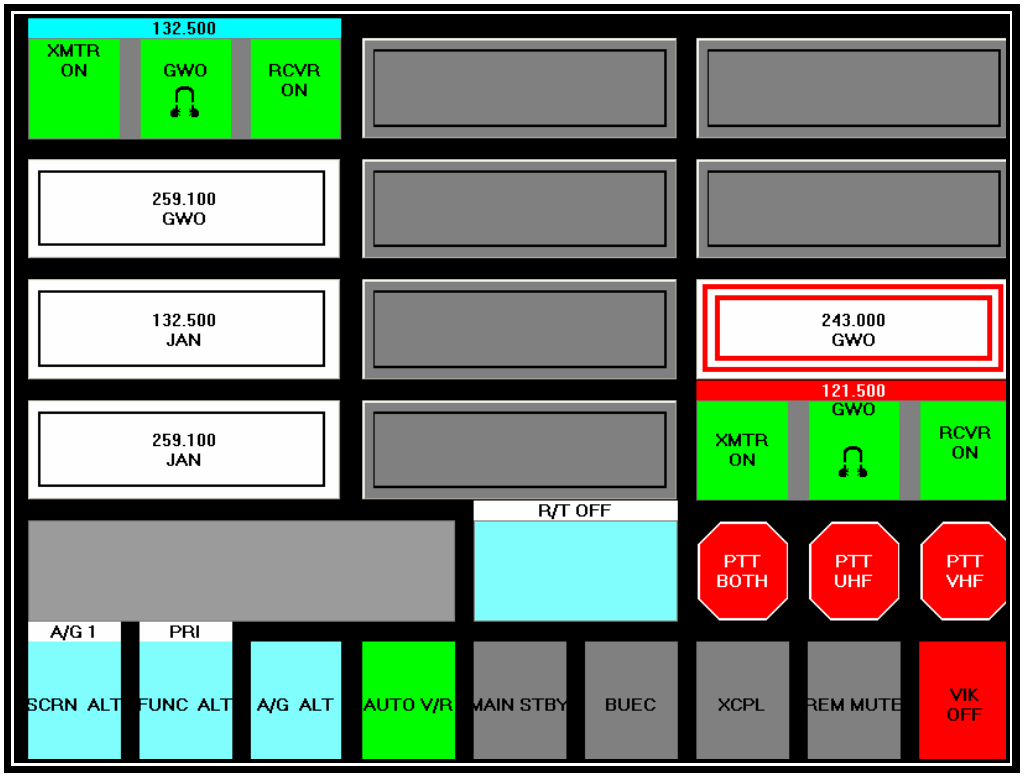


Figure 8-9 VCS Client A/G dialog Screen (1 of 2)

Some key A/G dialogue screen (AgScreenDlg class) methods include:

DSP_Status(CString State) – Called from the message handler upon receipt of a heartbeat with a "new" DSP status, this function either displays the "DSP Down – no audio available!" red banner or hides it.

EndMessage() - After the 30 second timer is run-down, this method removes the message from the message text area.

FindFrequency(CString FreqNum, CString Site, int &ScreenNum, int &Position)- FindFrequency() loops through both A/G1 and A/G2 screen buttons looking for a match to the given argument variables FrequNum and Site. If found this method returns pointers to the screen and position location of the button.

FuncScreenToPri(), FuncPriToScreen(), FuncScreenToAlt(), FuncAltToScreen() – These functions are called to hide and show the primary, screen, and alternate button sets, depending on what is currently displayed and which button set is desired.

InsertAgOneFrequency(int Position, CString Freq, CString nSite, CString FrqChannel, bool Emer, bool RM, bool BUEC, bool PTTP, bool GM, bool StbyX, bool StbyR, bool XC, bool GC, Direction XCDirec) – The InsertAgOneFrequency method adds a new frequency button to A/G screen 1.

InsertAgTwoFrequency(int Position, CString Freq, CString nSite, CString FrqChannel, bool Emer, bool RM, bool BUEC, bool PTTP, bool GM, bool StbyX, bool StbyR, bool XC, bool GC, Direction XCDirec) – The InsertAgTwoFrequency method adds a new frequency button to A/G screen 2.

OnClickPrifuncButton1() – When the user clicks the SCRN ALT button from an A/G screen, this function toggles the buttons along the bottom of the A/G screen between primary options buttons and screen select buttons. When the buttons are switched to the screen select buttons, the SCRN ALT button is displayed with a green background and flashing black text.

OnInitDialog() The OnInitDialog method initializes the A/G 1 and 2 dialogue screens.

OnTimer(UINT nTimerID) The OnTimer method performs a variety of functions at intervals specified with the SetTimer method. Some of these tasks include clearing messages, changing button options, and blinking buttons.

OnMouseDownAuxmessButton(short FAR* Button, short FAR* Shift, float FAR* X, float FAR* Y) – This function toggles all A/G incoming audio between the headset and the speakers. When RT is on all A/G audio is routed to the speakers and the frequency VR button is overridden.

OnMouseDownFreqButton1(short Button, short Shift, long X, long Y) - There are twelve OnMouseDownFreqButton functions, one for each A/G Screen Frequency button. When an unselected A/G frequency button is pressed, these functions call the

PushUnselectedFrequency(i) routine, where *"i"* represents the number of the associated A/G frequency button pushed.

OnMouseDownHslsBmp1(short Button, short Shift, long X, long Y) - There are twelve OnMouseHslsBmp functions., one for each A/G Screen Frequency button. When the bitmap layered on top of the voice router (VR) button of a selected frequency is pressed, the PushVoiceRouter() function is called to direct the incoming A/G audio to either the speakers or the headset for the selected frequency channel.

OnMouseDownRcvrButton1(short Button, short Shift, long x, long y) - There are twelve OnMouseDownRcvrButton functions, one for each A/G Screen Frequency button. When the reciever button of a selected frequency is pressed, the PushReceiver() function is called to set the desired state of the receiver for that frequency channel.

OnMouseDownScrfuncButton1(short Button, short Shift, long x, long y), OnMouseDownScrfuncButton2(short Button, short Shift, long x, long y), OnMouseDownScrfuncButton3(short Button, short Shift, long x, long y), OnMouseDownScrfuncButton4(short Button, short Shift, long x, long y), OnMouseDownScrfuncButton5(short Button, short Shift, long x, long y), OnMouseDownScrfuncButton6(short Button, short Shift, long x, long y) Subsequent to a SCRN ALT button press, the user selects one of the screen select buttons located at the bottom of the dialogue screen which calls on one of these corresponding functions to direct the application focus and to display the associated dialogue screen.

OnMouseDownVrButton1(short Button, short Shift, long x, long y) - There are twelve OnMouseDownVrButton functions, one for each A/G Screen Frequency button. When this voice router (VR) button of a selected frequency is pressed, the PushVoiceRouter() function is called to direct the incoming A/G audio to either the speakers or the headset for the selected frequency channel.

OnMouseDownXmtrButton1(short Button, short Shift, long X, long Y) - There are twelve OnMouseDownXmtrButton functions, one for each A/G Screen Frequency button. When the transmitter button of a selected frequency is pressed, the PushTransmitter() function is called to set the desired state of the transmitter for that frequency channel.

OnMouseDownPrimsgButton(short Button, short Shift, long X, long Y), OnMouseDownPrimsgText(short FAR* Button, short FAR* Shift, float FAR* X, float FAR* Y) – These methods detect that the user intends to clear the primary message box and consequently call the EndMessage routine.

OnMouseDownPttuhfButton(short Button, short Shift, long x, long y), OnMouseDownPttuhfvhfButton(short Button, short Shift, long x, long y), OnMouseDownPttvhfButton(short Button, short Shift, long x, long y) – When any of the non-latching emergency PTT buttons are pressed, a message is sent to the COM Server indicating that the user intends to transmit on all A/G frequency channels, enabling PTT.

OnMouseDownVikButton2(short Button, short Shift, long X, long Y) – Depending on the state of the software VIK when the VIK button is pressed, this routine presents or hides the software VIK.

OnMouseUpPttuhfButton(short Button, short Shift, long x, long y), OnMouseUpPttuhfvhfButton(short Button, short Shift, long x, long y), OnMouseUpPttvhfButton(short Button, short Shift, long x, long y) – When any of the non-latching emergency PTT buttons are released, a message is sent to COM sever to disable emergency PTT for A/G transmission.

PTTBegin(Cstring Freq_Channel), PTTEnd(CString Freq_Channel) – These methods loop through all A/G frequency buttons upon a receipt of a PTTBegin or PTTEnd message, changing the color of the selected frequencies XMTR buttons. This is an indication that the user is transmitting on that frequency channel.

PTTNotify(CString State, CString Freq_Channel) – Called from the message handler (OnReceive) upon receipt of a PTTNotify Message from the COM Server. This method loops through all A/G frequency buttons changing the color of the selected frequency RCVR buttons. This is an indication that another user within the training sector is transmitting on the frequency channel associated with the changed RCVR buttons.

PushReceiver(int Position, CfpBtn* ThisButton) - Based on the current state of the receiver for a given frequency channel, this routine enables or disables the ability to receive incoming A/G audio and then refreshes the frequency receiver button with the appropriate button text (ie. “ON” or “OFF”). The Position argument represents the frequency button position of the receiver button pushed and the ThisButton argument provides a handle to the actual receiver button object. This method may also call a MessageHandler Method to submit a message to the COM Server indicating an A/G frequency state change.

PushTransmitter(int Position, CfpBtn* ThisButton) – Based on the current state of the transmitter for a given frequencychannel, this routine enables or disables the ability to transmit A/G audio and then refreshes the transmitter frequency button with the appropriate text (ie. “ON” or “OFF”). The Position argument represents the frequency button position transmitter of the button pushed and the ThisButton argument provides a handle to the actual transmitter button object. This method may also call a MessageHandler Method to submit a message to the COM Server indicating an A/G frequency state change.

PushUnselectedFreq(int Position) – Turns on the transmitter and receiver for a given frequency and refreshes the A/G frequency buttons. This method may also call a MessageHandler Method to submit a message to the COM Server indicating an A/G frequency state change.

PushVoiceRouter(int Position, CfpBtn* ThisButton) - Based on the current routing of the incoming A/G audio for a given frequency channel, this routine directs the incoming A/G audio to either the speakers or the headset for the selected frequency channel. The method then refreshes the VR frequency button with the appropriate speaker or headset

icon. The Position argument represents the frequency button position of the VR button pushed and the ThisButton argument provides a handle to the actual VR button object. This method may also call a MessageHandler Method to submit a message to the COM Server indicating an A/G audio routing state change.

RefreshButtons() – This method refreshes all frequency buttons on the "Active" (currently display) A/G screen.

Ring(int ScreenNum – This method is called when an incoming call is being received on a G/G screen. It will alert the user to the call and flashing the screen alt button and the corresponding G/G screen select button.

RingOff(int ScreenNum) – This routine is called to return the screen alt and G/G screen select buttons to a non-flashing state after an incoming call has been alerted to the user.

SetAssociations(CAgScreenDlg* pAG, GgScreen* pGG, UtilityScreen* pUTL, AgStatus* pAGS) SetAssociations is a scoping method which creates pointer variables to all other dialogue screens and the message handler, giving visibility to these object classes.

SendWarnMsg(CString Message), SendErrorMsg(CString Message), SendSystemMsg(CString Message) - Displays a warning message passed in as the Message string argument and also displays the appropriate warning ,error, or system icon in the in the Message Text Area.

SetActiveScreen(int NewScreen) – This routine sets the ActiveScreen variable prior to a screen refresh. ActiveScreen is an integer variable with a value of either a 1 or a 2 representing the appropriate A/G screen (ie. A/G1 or A/G2).

SetRT(bool State) – Called when the AuxMessage box is pressed, this method sets the state RT state for incoming A/G audio. This method may also call a MessageHandler Method to submit a message to the COM Server indicating a Radio Transmission state change.

UpdateVikButton(CString State) – This function presents the VIK button on the A/G screen with the appropriate color and text. The button indicates the VIKs current state of either on or off.

UpdateMonList(CString MonListText) – This function is called by message handler to update the AUX Msg box when an OVR notification (begin or end) from another position is received. When an OVR call is in progress the AUX Msg box displays the position ID of an Overriding sector.

8.2.2.4 Ground to Ground Screen (GgScreen.cpp)

The Ground to Ground dialogue screen (G/G) allows the VCS client users to initiate, terminate and receive various types of G/G calls that simulate sector-to-sector and interfacility communications. Call types include IP (Holler, Ring, and Dial), IC (Intercom), OVR (Override), and VMON (voice monitor) calls.

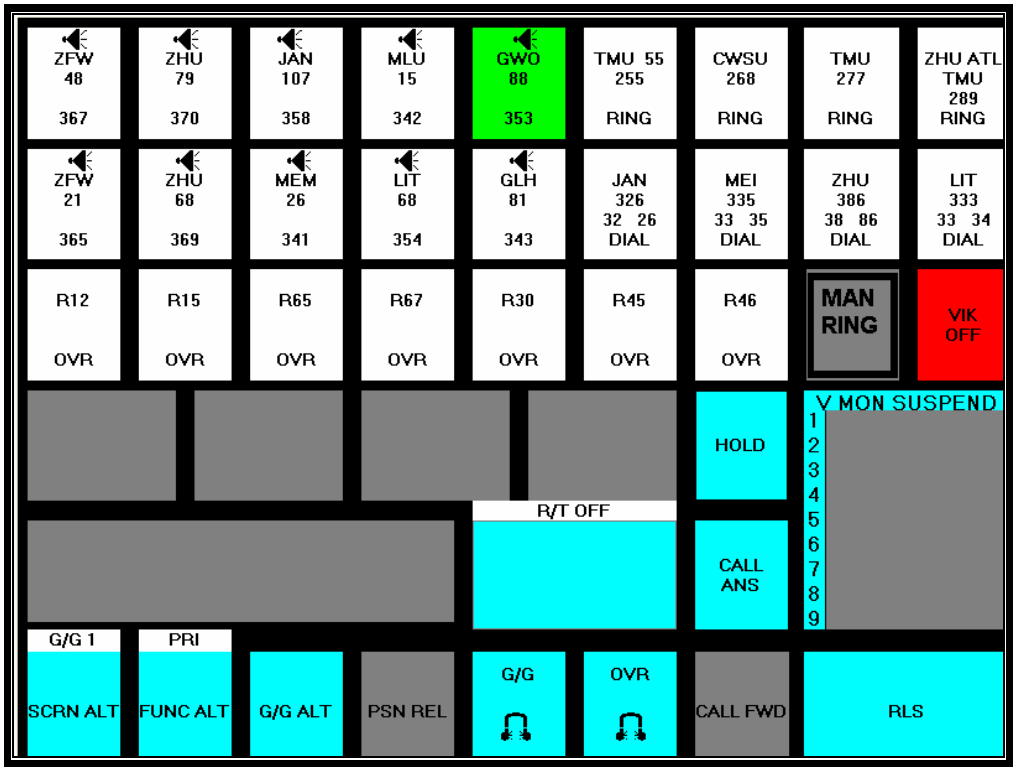


Figure 8-10 VCS Client G/G dialog Screen (1 of 2)

Some common G/G dialogue screen (GgScreen class) methods include:

CaButtonPushed(int Position) - Detects that a CA button has been pushed. This routine evaluates the selected button's state and then performs the actions necessary to carry out the communication request intended by the user.

CallAnswered(CString ID, CString DestId, CallType Type, CString TrunkId) – This method is called from the message handler. It attempts to locate a G/G button in the ringback state associated with an answered call.

CallReceived(CString ID, CString SrcId, CallType Type, CString TrunkId) – This method is called from the message handler. It attempts to locate a G/G button associated

with an incoming call. If the call is located the button's state is set to ringing and the buttons are refreshed.

CallReleased(CString ID, CString DestId, CallType Type, CString TrunkId) – This method uses the FindCall method to locate a specific incoming call. If the call is found it performs the appropriate actions associated with releasing the call depending on the call's current state.

DaButtonPushed(int Position) – Detects that a DA button has been pushed. This routine evaluates the selected button's state and call-type and then performs the actions necessary to carry out the communication request intended by the user. Depending on the current state and type of the button pressed, this method may initiate or release a call, join or leave a call, or display the VIK to complete a dial call.

DaButtonReleased(int Position) – When a non-latching button is released this method terminates the call and if necessary resumes any suspended voice monitoring.

DSP_Status(CString State) – Called from the message handler upon receipt of a heartbeat with a "new" DSP status, this function either displays or hides the "DSP Down – no audio available!" red banner.

EndMessage() - After a 30 second timer expires, this method removes the message from the message text area.

FindCall(CString ID, CString SrcId, CallType Type, CString TrunkId, int& ScreenNum, int& Position) - FindCall loops through both G/G screens buttons looking for a match of the given argument variables. If found this method returns pointers to the screen and position location of the button.

FuncScreenToPri(), FuncPriToScreen(), FuncScreenToAlt(), FuncAltToScreen() – These functions are called to hide and show the primary, screen, and alternate button sets, depending on what is currently displayed and which button set is desired.

InsertGgOneCall(CString NewID, CString NewSrcId, int Position, CString NewLabel, bool IsLatching, CallType NewCallType, bool HollerOn, bool DialOn, CString TrunkId), InsertGgTwoCall(CString NewID, CString NewSrcId, int Position, CString NewLabel, bool IsLatching, CallType NewCallType, bool HollerOn, bool DialOn, CString TrunkId) - These functions create a new G/G call button from the information provided in the parameters and insert the button into the appropriate position on the proper screen.

InUseCallInProgress(CString ID, CString DestId, CallType Type, int State, CString TrunkId) – This method uses the FindCall method to locate all IP call buttons associated with a given TrunkId. Depending on the return value of FindCall and this method's State argument, the state of the buttons are set to either INUSECIP (In Use Call In Progress) or IDLE, and the buttons are refreshed. A green call-in-progress bar is placed at the top of each associated button in the INUSECIP state. For all associated buttons set to the IDLE state the green bar is removed.

InUseCallPlaced(CString ID, CString DestId, CallType Type, int State, CString TrunkId) - This method uses the FindCall method to locate all IP call buttons associated with a given TrunkId. Depending on the return value of FindCall and this method's State argument, the state of the buttons are set to either INUSECP (In Use Call Placed) or IDLE, and the buttons are refreshed. An amber call-placed bar is placed at the top of each associated button in the INUSECP state and the button is disabled for use. For all associated buttons set to the IDLE state the amber bar is removed and the button is enabled.

OnInitDialog() The OnInitDialog method initializes the G/G 1 and G/G 2 dialogue screens.

OnMouseDownCaButton1(short Button, short Shift, long x, long y), OnMouseDownCaLabel1(short FAR* Button, short FAR* Shift, float FAR* X, float FAR* Y) – These functions detect that a CA button has been pushed, and subsequently call the CaButtonPushed method.

OnMouseDownCallansButton(short Button, short Shift, long X, long Y) - Answers any incoming DA or CA call and sets the call state to active.

OnMouseDownDaButton1(short Button, short Shift, long X, long Y) – There are twenty five of these OnMouseDownDaButton methods. One associated with each DA button position. They are named "OnMouseDownDaButton1" through "OnMouseDownDaButton25" respectively. These methods are called when a DA button is pressed. The methods subsequently call the DaButtonPushed method.

OnMouseDownGgaltfuncButton1(short Button, short Shift, long x, long y) – Detects that the Holler on/off button has been pushed. This button is currently disabled.

OnMouseDownGgAuxmsgButton(short FAR* Button, short FAR* Shift, float FAR* X, float FAR* Y) - This function toggles all A/G incoming audio between the headset and the speakers.

OnMouseDownGgprifuncButton3(short Button, short Shift, long x, long y) – Detects that the user clicked on the G/G ALT function button. This function sets the ActiveScreen variable and calls the appropriate method to toggle the focus between the GG/1 and G/G2 dialogue screens.

OnMouseDownGgprifuncButton7(short Button, short Shift, long x, long y) - Detects that the Call forward button has been pushed. This button is currently disabled.

OnMouseDownGgprifuncButton1(short Button, short Shift, long x, long y) - When the user clicks the SCRN ALT button from a G/G screen, this function toggles the buttons along the bottom of the A/G screen between primary options buttons and screen select buttons. When the buttons are switched to the screen select buttons, the SCRN ALT button is display with a green background and flashing black text.

OnMouseDownGgprifuncButton2(short Button, short Shift, long x, long y) – This function is called when a user presses the FUNCT ALT button toggling between the primary and alternate G/G functions.

OnMouseDownGgprifuncButton8(short Button, short Shift, long X, long Y) – Calls the ReleaseCurrentCall method to release all currently active calls and if necessary it will resume any Monitor call types that were previously suspended die to an activated G/G call.

OnMouseDownGgPrimsgButton(short FAR* Button, short FAR* Shift, float FAR* X, float FAR* Y), OnMouseDownGgPrimsgText(short FAR* Button, short FAR* Shift, float FAR* X, float FAR* Y) - These methods detect that the user intends to clear the primary message box and consequently call the EndMessage routine.

OnMouseDownGgscrnfuncButton1(short Button, short Shift, long x, long y) The are six MouseDownGgcrFuncButton functions named "OnMouseDownGgscrfuncButton*1*" through "OnMouseDownGgscrfuncButton6". Subsequent to a SCRN ALT button press, the user may select one of the screen buttons 1 through 6 located at the bottom of the dialogue screen, which calls on one of these corresponding functions to direct the focus to and display the associated dialogue screen.

OnMouseDownVikButton(short Button, short Shift, long X, long Y) – Called when the VIK button from either G/G screen is pressed. This method calls the VikControl method to either display or hide the VIK. When the VIK is hidden this method also attempts to clean up any uncompleted calls.

OnMouseUpDaButton1(short Button, short Shift, long X, long Y) - There are twenty five of these OnMouseUpDaButton methods. One associated with each DA button position. They are named "OnMouseUpDaButton1" through "OnMouseDownDaButton25" respectively. These methods are called when a DA button is released and consequently call the DaButtonReleased method.

OnTimer(UINT nTimerID) - The OnTimer method performs a variety of functions at intervals specified with the SetTimer method. Some of these tasks include clearing messages, changing button options, and blinking buttons.

ProcessVik(CString ID, CString DestId, CallType Type, CString VikSwitch) - Function is called by the VIK dialogue screen to process all indirect access requests.

SendWarnMsg(CString Message), SendErrorMsg(CString Message), SendSystemMsg(CString Message) - Displays a warning message specified by the Message string argument. Also displays the appropriate warning , error, or system icon in the in the Message Text Area.

SetAssociations(CAgScreenDlg* NewAg, AgStatus* NewStat, UtilityScreen* NewUtil, MessageHandler* NewMsgH, VikScreen* NewVikScreen) - SetAssociations is a method which creates pointer variables to all other dialogue screen objects. This gives the G/G dialogue screen visibility and access to the other dialogue screen public objects as well as the Message Handler public methods.

SetRT(bool State) – Called when the AuxMessage box is pressed, this method sets the state RT state for incoming A/G audio. This method may also call a MessageHandler Method to submit a message to the COM Server indicating a Radio Transmission state change.

ShowGgOne(), ShowGgTwo()- These functions refresh all of the DA buttons to present the G/G1 or G/G2 information and sets the ActiveScreen to G/G1 or G/G2.

ReleaseCurrentCall() – This method loops through all the G/G DA and CA call buttons. If a G/G call button is found in the active or ringback state a message is sent to the COM Server, via the MessageHandler, releasing the call. The button is then returned to its idle state.

UpdateMonList(CString MonListText) – This function is called by message handler to update the AUX Msg box when an OVR notification (begin or end) from another position is received. When an OVR call is in progress the AUX Msg box displays the position ID of an Overriding sector.

VikControl(CString Action) – Presents or hides the software VIK to the user and updates the VIK button with the appropriate color and text.

VoiceMonRelease(CString ID, CString DestId, CallType Type) – Called by the message handler when a voice monitor call has been released via the VIK dialogue screen.

8.2.2.5 Air to Ground Status Screen (AgStat.cpp)

The Air to Ground Status (A/G STAT) Dialogue Screen (see Figure 8-11) is a summary of the A/G1 and A/G2 screens, allowing the VCS client to simultaneously view the current state of all adapted air to ground frequencies. Emergency PTT and RT functions are also available from this dialogue screen.

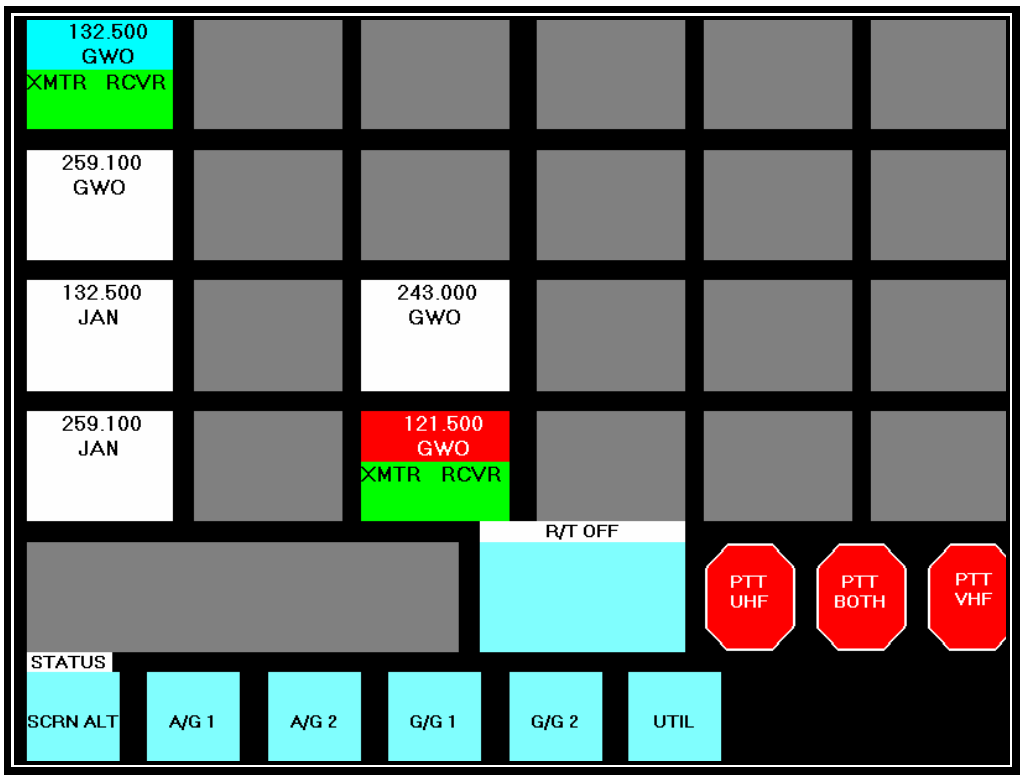


Figure 8-11 VCS Client A/G Status Screen

Some common A/G STAT dialogue screen (AgStat class) methods include:

AddFrequency(int Position, Frequency *NewFrequency) Called by the AddFrequency method from the A/G screen dialogue object class, the AddFrequency method presents all adapted A/G frequencies.

DSP_Status(CString State) Called from the message handler upon receipt of a heartbeat with a "new" DSP status, this function either displays or hides the "DSP Down – no audio available!" red banner.

EnableEPTT() Upon initialization the Emergency PTT buttons are disabled. This method enables the emergency PTT buttons when first emergency frequency is adapted. If no emergency frequency is adapted the emergency PTT buttons remain disabled.

OnInitDialog() The OnInitDialog method initializes the A/G 1 and 2 dialogue screens.

OnMouseDownStatAltscreenButton2(short Button, short Shift, long X, long Y), OnMouseDownStatAltscreenButton3(short Button, short Shift, long X, long Y), OnMouseDownStatAltscreenButton4(short Button, short Shift, long X, long Y), OnMouseDownStatAltscreenButton5(short Button, short Shift, long X, long Y), OnMouseDownStatAltscreenButton6(short Button, short Shift, long X, long Y) These methods are called when a user presses a screen button located at the bottom of the A/G Status screen, allowing the user to switch to the A/G1, A/G2, G/G1, G/G2, or UTIL dialogue screens respectively.

OnMouseDownStatAuxmsg(short FAR* Button, short FAR* Shift, float FAR* X, float FAR* Y) This function toggles all A/G incoming audio between the headset and the speakers. When RT is on, all A/G audio is routed to the speakers and the frequency VR button is overridden.

OnMouseDownStatUhfButton(short Button, short Shift, long X, long Y), OnMouseDownStatUhfvhfButton(short Button, short Shift, long X, long Y), OnMouseDownStatVhfButton(short Button, short Shift, long X, long Y) When any of the non-latching emergency PTT buttons are pressed, a message is sent to the COM server indicating that the user intends to transmit on all A/G frequency channels, enabling PTT.

OnMouseUpStatUhfButton(short Button, short Shift, long X, long Y), OnMouseUpStatUhfvhfButton(short Button, short Shift, long X, long Y), OnMouseUpStatVhfButton(short Button, short Shift, long X, long Y) When any of the non-latching emergency PTT buttons are released, a message is sent to the COM server to disable emergency PTT for A/G transmission.

RefreshButtons() This method refreshes all frequency buttons on the A/G Status screen.

SetAssociations(CAgScreenDlg* NewAg, UtilityScreen* void NewUtil, GgScreen* NewGg, MessageHandler* NewMsgH) SetAssociations is a scoping method which creates pointer variables to all other dialogue screens and the message handler, giving visibility to these object classes.

SetRT(bool State) Called when the AuxMessage box is pressed, this method sets the RT state for incoming A/G audio. This method may also call a MessageHandler Method to submit a message to the COM Server indicating a Radio Transmission state change.

8.2.2.6 Utility Screen (UtilScreen.cpp)

The Utility screen displays the client machine's physical and logical position Ids in the bottom right-hand corner. This dialogue screen also allows the user to adjust the volume of the headset and loudspeaker audio.

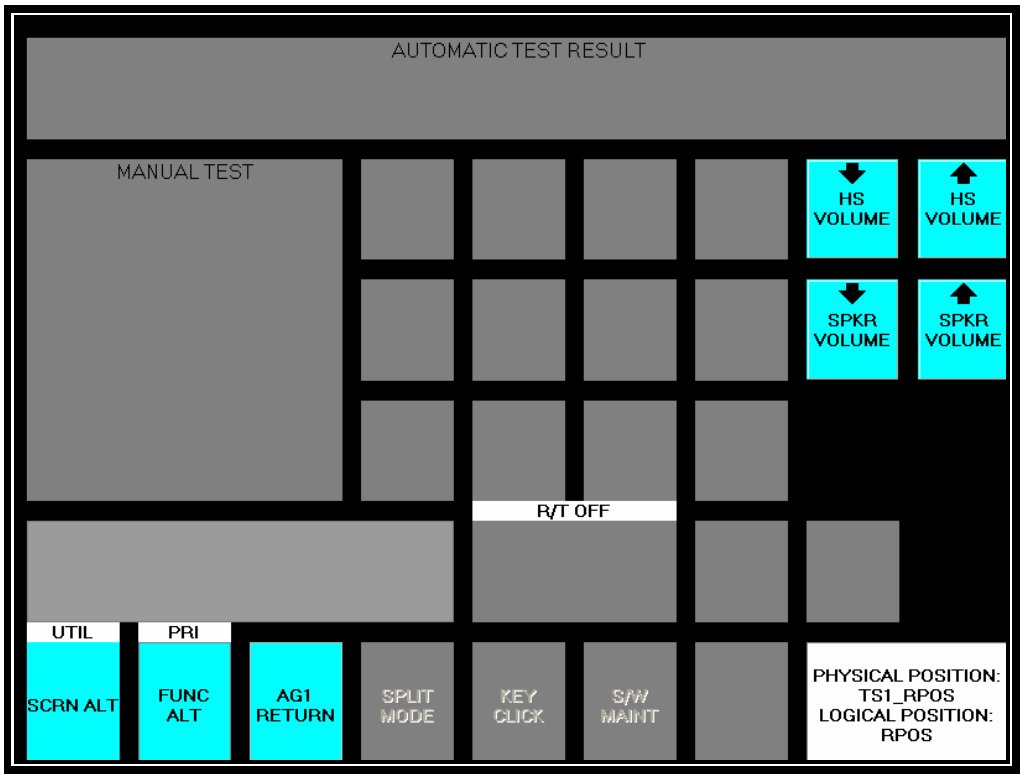


Figure 8-11 VCS Client Utility Screen

DSP_Status(CString State) Called from the message handler upon receipt of a heartbeat with a "new" DSP status, this function either displays the "DSP Down – no audio available!" red banner or hides it.

OnInitDialog() The OnInitDialog method initializes the Utility dialogue screen .

OnMouseDownUtilScrnalt(short Button, short Shift, long x, long y) When the user clicks the SCRN ALT button this function toggles the buttons along the bottom of the A/G screen between primary options buttons and screen select buttons. When the buttons are switched to the screen select buttons, the SCRN ALT button is displayed with a green background and flashing black text.

OnMouseDownUtilScrfuncButton1(short Button, short Shift, long x, long y),
OnMouseDownUtilScrfuncButton2(short Button, short Shift, long x, long y),
OnMouseDownUtilScrfuncButton3(short Button, short Shift, long x, long y),
OnMouseDownUtilScrfuncButton4(short Button, short Shift, long x, long y),
OnMouseDownUtilScrfuncButton5(short Button, short Shift, long x, long y),
OnMouseDownUtilScrfuncButton6(short Button, short Shift, long x, long y)

Subsequent to a SCRN ALT button press, the user selects one of the screen select buttons located at the bottom of the dialogue screen calling on one of these corresponding functions to direct the application focus to and displays the associated dialogue screen.

OnMouseDownUtilScrnrtrn(short Button, short Shift, float X, float Y) When the screen return button is pressed, this routine switches the client to the previous dialogue screen represented in the Utility Object by the PreviousScreen variable.

OnMouseDownVolumedownButton(short Button, short Shift, long X, long Y),
OnMouseDownVolumeupButton(short Button, short Shift, long X, long Y),
OnMouseDownSpkvoldownButton(short Button, short Shift, long X, long Y),
OnMouseDownSpkvolupButton(short Button, short Shift, long X, long Y) These functions send a message to the COM server indicating the user's intent to adjust the volume level of all incoming audio directed to the loudspeaker or the headset.

PriToScreen(), ScreenToPri() These functions are called to hide and show the primary and screen button sets.

SetAssociations(CAgScreenDlg* NewAg, AgStatus* NewAgStat, GgScreen* NewGg, MessageHandler* NewMsgHand) SetAssociations is a method which creates pointer variables to all other dialogue screen public objects and methods. Also provides visibility to the MessageHandler class.

SetPsn(CString ID, CString ComputerName) This method sets the Logical Position ID on the Utility screen.

SetLastScreen(CString LastScreen) This function sets the PreviousScreen variable to the value of the given LastScreen argument.

8.2.2.7 VIK Screen (VikScreen.cpp)

The VIK dialogue screen is a software version of the VSCS Indirect Access Keyboard (VIK). It is presented to the user by either selecting a dial type G/G IP DA button or by pressing the VIK button located on the any of the A/G or G/G dialogue screens. When initiated, the user may use the VIK keypad to enter dial codes to initiate a call. The VIK may also be used to release G/G calls.

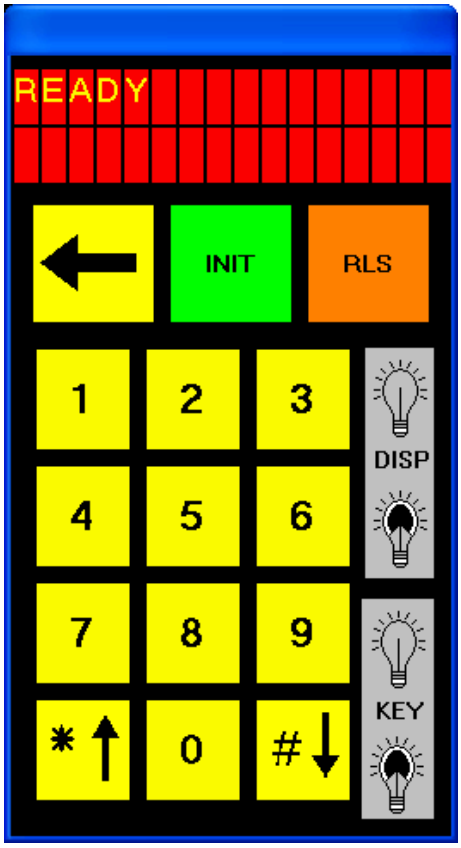


Figure 8-12 VCS Client VIK Dialog Screen

Some common A/G dialogue screen (AgScreenDlg class) methods include:

ActivateDisplay(), DeactivateDisplay() These routines brighten or darken the background color of the VIK display. The DeactivateDisplay routine also blanks out the message array variables and calls UpdateDisplay to clear any message previously displayed.

ActivateKeys(), DeactivateKeys() These routines brighten or darken the background color of the keypad keys and enables or diables them for use.

Clear() This routine clears the VIK display and deactivates the VIK.

InitiateCall(CallType Type) As the user presses keypad keys, the VIK evaluates what the user's intent is. When enough input is received to determine that user's intent, this routine is called to display preliminary messages on the VIK. (ie. Keypad "8" is pressed – This routine displays "OVR CALL # _")

InitiateSpecialFunction(SpFuncType Type) Currently voice monitoring (VMON) is the only special function enabled and disabled through this routine in the VIK object class.

KeyPressed(char ThisKey) Each time a key pad key is pressed a message is sent to the COM server triggering an audible keytone associated with the key pressed. The KeyPressed routine also evaluates and processes the user input each time a keypad key is pressed. Upon successful completion of a valid G/G indirect access call or special function the appropriate function calls are made to facilitate the user request.

KeyReleased(char ThisKey) This method sends a message to the COM server to discontinue the audible keytone.

OnMouseDownVikBkspceKey(short Button, short Shift, long X, long Y) When the user presses the backspace key, represented by the left-pointing arrow, this routine clears the last most type character inputted by the user.

OnMouseDownVikInitKey(short Button, short Shift, long X, long Y) When a user presses the Init key on the VIK this routine initializes the VIK for user input by activating the VIK display and keypad.

OnMouseDownVikRlsKey(short Button, short Shift, long X, long Y) This method deactivates the display and the keypad, then calls the GgScreen object class ProcessVik() routine to release the current call.

OnMouseUpNumberKey1(short Button, short Shift, long X, long Y) - OnMouseUpNumberKey11(short Button, short Shift, long X, long Y) When a user releases a VIK keypad key, one of these corresponding methods make a call to the KeyReleased function passing the appropriate argument representing the keypad released.

OnInitDialog() The OnInitDialog method initializes the VIK dialogue screen.

OnTimer(UINT nTimerID) The OnTimer method performs a variety of functions at intervals specified with the SetTimer method. Some of these tasks include clearing messages and hiding the VIK.

SetAssociation(GgScreen* GG, MessageHandler* MH, CAgScreenDlg* AG) SetAssociations is a scoping method which creates pointer variables to all other dialogue screens and the message handler, giving visibility to these object classes.

SendMsg(CString Msg) This function activates the VIK display, presents the message given by the Msg argument by calling UpdateDisplay(), deactivates the keypad, then initiates a timer to clear the message.

SwitchScreens() This method toggles the focus of the VCS application between the G/G and A/G screens.

UpdateDisplay() Loops through the characters of the CString text of the VIK Object variables MsgOne and MsgTwo and places them in the VIK Object character arrays LineOne and LineTwo for presentation on the VIK

VikIpCall() This routine initializes the VIK and sets the call type to IP when a G/G dial DA button is pressed.

8.3 COTS software requirements

The VCS COM Server, Clients, and Adaptation Builder applications all use COTS Active X plug-in objects within their application user interface. These objects use the standard Active X object format and the objects are represented by files with an ".ocx" suffix in the Windows XP System32 folder. These files are installed in the appropriate place during VCS Setup. The objects are pulled in to each software project using the Add/Remove Active X component feature within the Visual Studio IDE.

8.3.1 COM Server COTS Software

The following Active X objects are used in the COM Server application:

- Global Majic Software (GMS) Toggle object (Toggle.ocx)
- GMS Slider Object (Slider.ocx)
- Impulse Studio Animated Label Objects (ImpulseAniLabel.ocx)

8.3.2 VCS Client COTS Software

The following Active X objects are used in the VCS Client Application:

- Far Point (fpBtn) Button Objects (btn32a20.ocx)
- Impulse Studio Animated Label Objects (ImpulseAniLabel.ocx)

8.3.3 VCS Adaptation Builder COTS Software

The following Active X objects are used in the VCS Adaptation Builder Application:

- Far Point (fpBtn) Button Objects (btn32a20.ocx)

8.4 COM Server Signal RPC Client Software

The Signal RPC Software was provided by the MMAC Signal software group. The decision to incorporate it into the IATS VCS for the purpose of remotely controlling networked clients was prudent since Signal software engineers are already familiar with it. The Signal RPC software is a separate program written in C++ that comprises a server and client component (executable). The client component resides on the IATS COM

Server. The server component resides on each of the IATS Clients and the DSPs. The IATS COM Server utilizes the RPC client component as necessary to send directives to the specific IATS Client's RPC server component. The IATS Client's and IATS DSP's RPC server component runs as a background process and is auto-started when the machine boots up.

8.4.1 Signal RPC Client Software

The IATS COM Server takes advantage of the Signal RPC Client application's ability to communicate with a peer RPC Server application on other networked computers. The IATS COM Server calls the locally resident Signal RPC Client application, providing it a hostname and command value. The hostname can be any other networked computer that is running the RPC Server. The command values are as follows.

Command Value	Purpose
1	Start the AgScreen.exe executable on the specified client machine.
2	Reboot the specified client machine.
3	Create a "public" user account on the specified networked machine.
4	Start the IATS-VCS.vne executable on the specified DSP machine.

8.4.2 Signal RPC Server Software

The IATS Clients and DSP take advantage of the Signal RPC Server application's ability to communicate with a peer RPC Client application on the IATS COM Server. The RPC Server application running on the IATS Client and DSP receives a directive from the IATS COM Server's RPC Client application. The directive will specify whether to start the IATS Client software or reboot the computer. For the RPC Server application running on the DSP, the directives are to start the IATS DSP software or reboot the computer.

8.5 VCS DSP Logic Description

The DSP is the IATS VCS component that makes the connection between physical input and output audio ports to enable/disable the proper communication paths. The DSP also plays special tones that the VCS Client users hear (e.g. dial, ring, error, etc.). The DSP responds to IATS COM Server signals to establish or break specified communication paths allowing voice and/or tones from the specified source to be output at the destination, typically an ATC headset or Display System Replacement (DSR) position console speaker. The DSP also actively sends "heartbeat" indications to the IATS COM Server to inform it that the DSP is still an active member of the IATS VCS.

The IATS DSP subsystem is COTS hardware and software that has been programmed to

listen to the IATS VCS COM Server. The DSP listens for a UDP data buffer that will direct it to establish or remove audio connections within its hardware platform. The UDP buffer is described in detail in Section 8.6. The DSP design pages contain the low-level logic to determine which positions are able to communicate. It is the VCS COM Server that determines when those connections are to be made and broken. See Section 8.5.4 for further descriptions of logic details.

8.5.1 V+ Introduction

The programming method used in the DSP is called the Visual Programming Language (VPL or V+) because it entails the creation of logic flow diagrams from a COTS symbol base. There are no typical programming language constructs, only objects (logic components) dragged and dropped onto a design sheet (.des file) and then connected (or "sewn") together by virtual threading.

8.5.2 V+ Visual Programming System Editor

The V+ editor platform allows compilation of objects onto design sheets. The tools available in the editor are: select, zoom, annotate, net, play & stop, object menu, port menu and project manager (shown in Figure 8-13). These menus are described in the V+ User Manual.

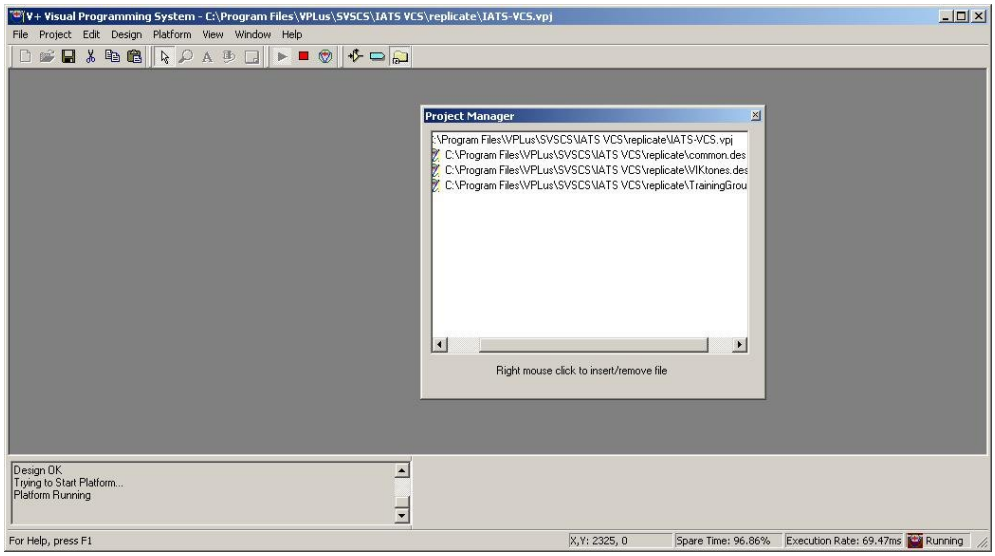


Figure 8-13 V+ Visual Programming Language Application

Caution: Currently (Ver 4.1.0), there is no UNDO in V+. The Editor can sometimes experience program hangs - Save often.

Multiple design sheets can be grouped into a project (.vpj file shown above). A project or a single sheet must be RUN in an environment (.vne file) for the hardware to be exercised. The design sheet is compiled just prior to running and represents the executable. When the DSP is executing a design sheet, all of the objects on the form are outlined in green (if they are shown) and a message displayed in the Status Area that indicates "Platform Running".

Caution: When making changes in the editor, the user must SAVE and RUN in the Programming environment. Then, while running, switch to the Run Time Environment (RTE), STOP and SAVE in the RTE for the changes to be propagated to the .vne file.

8.5.3 VCS V+ Run-Time Environment

The Run Time System (Figure 8-14) allows environment configuration (eg. UDP buffer size and port ID) and compiles all components from all sheets.

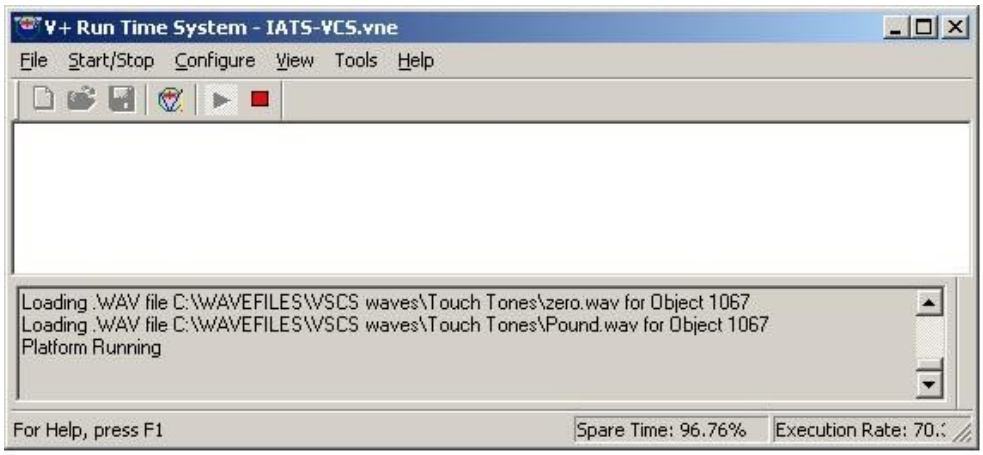


Figure 8-14 V+ Run Time Dialog

Configuring a vne file must be done when the platform is not running. The IO Devices tab contains options for UDP ports and the audio system channels as shown in Figure 8-15.

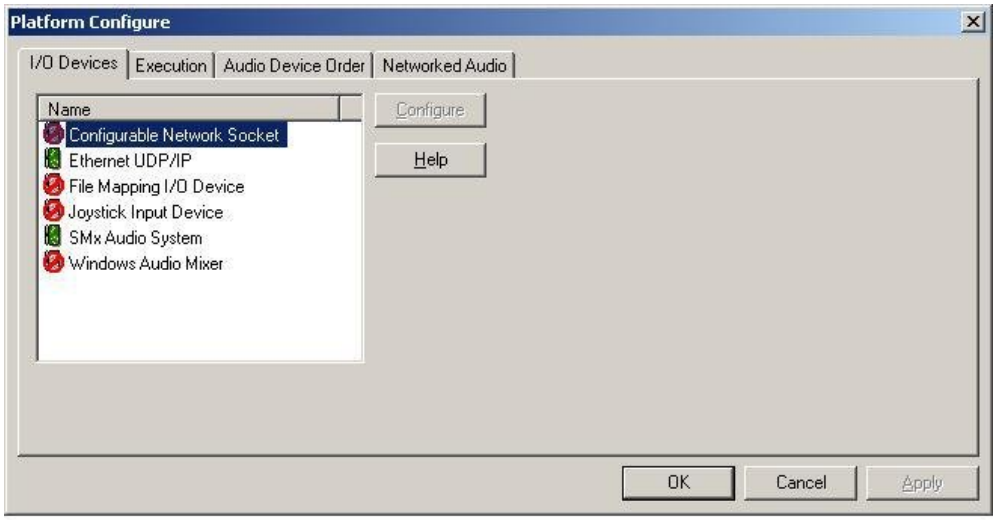


Figure 8-15 V+ Platform Configuration Dialog – I/O Devices

The user can configure the IO devices, the execution rate and other parameters. Figure 8-16 depicts examples of the current execution rate and UDP Output port definitions.

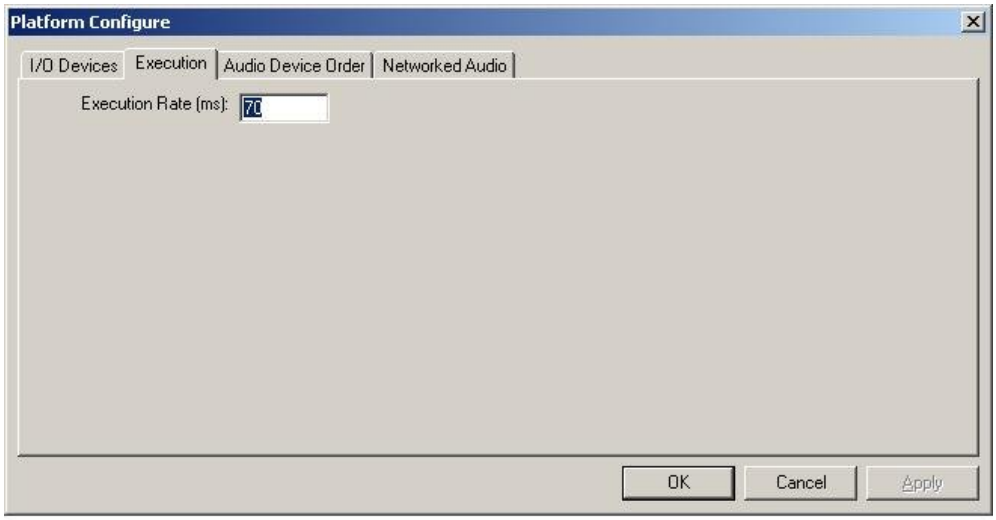


Figure 8-16 V+ Platform Configuration Dialog - Execution

Figure 8-17 displays the configuration settings for the UDP output packets used for the DSP heartbeat.

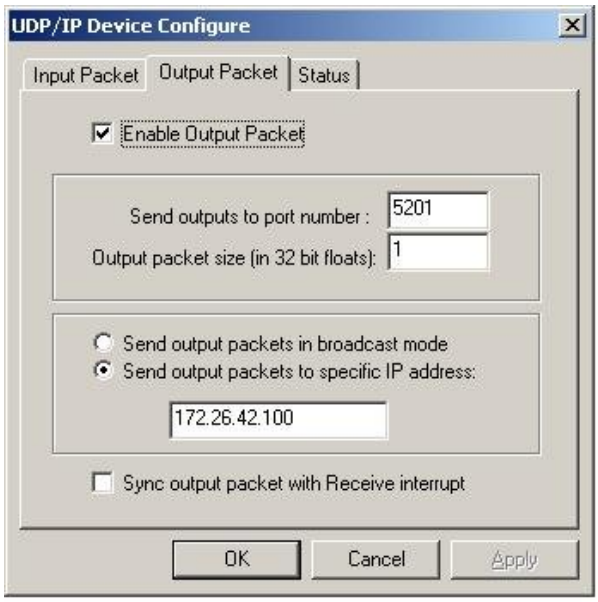


Figure 8-17 V+ UDP/IP Device Configuration – Output Packet

Listed below are the UDP/IP input and output port numbers for each of the IATS VCS DSPs:

DSP #	Input	Output
DSP1	**5101**	**5201**
DSP2	**5102**	**5202**
DSP3	**5103**	**5203**

The above numbers are not to be confused with the V+ UDP port numbers on the design sheets themselves. A V+ UDP port represents one 32 bit word inside the packet whose size is specified by the programmer. Figure 8-18 depicts the configuration settings for the

UDP input packets used for the DSP to Server communication in the IATS VCS. Note the size of the packet is the number of V+ ports available to the description sheets.

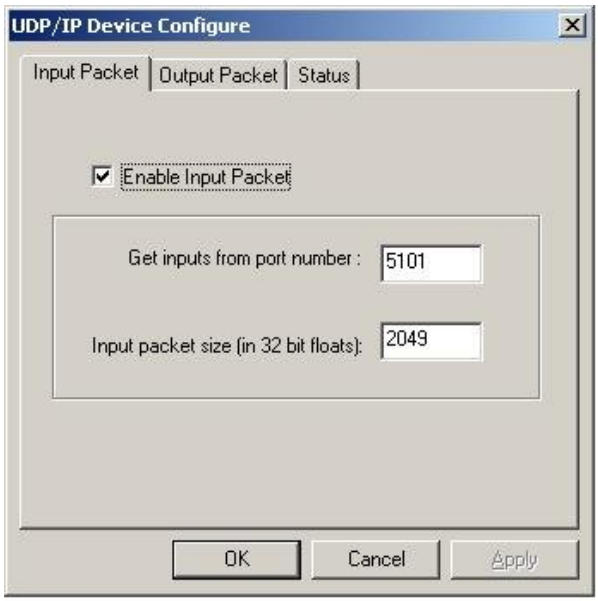


Figure 8-18 V+ UDP/IP Device Configuration – Input Packet

See V+ User Manual for more info.

8.5.4 VCS Design Worksheets

The VCS V+ design is broken up into several data sheets. Two sheets (**Common.des** & **VIK Tones.des**) provide logic common to all training sectors. These sheets contain the wave file player objects for all tones used in the DSP, the Master Instructor broadcast logic and the heartbeat signal sent to the Server.

There are also four sheets (**TrainingGroupx.des** where x= A,B, C, or D) that contain logic for the individual positions in a given sector including the students, pilots and instructors. The sheets also contain frequency logic, Volume and Gain levels, Monitor and Dial, Ring and Error Tone logic. A Training Group is a set of Training Sectors that use the same UDP port numbers on different DSPs. Since a lab may have up to 3 DSPs, a group can have up to 3 sectors. For example, the same TrainingGroupA sheets loaded on the three different DSPs will control Training Sector 1 on DSP1, Training Sector 5 on DSP2 and Training Sector 9 on DSP3.

The VCS design sheets use relatively few component types. Some components have Static Data associated with the object. The major components used in VCS DSP and their functions are:

Input and Output UDP ports – method used to transport data to the DSP from the VCS Server and heartbeats to the Server from the DSP. These are configured in the Run Time System - .vne file.

Analog output ports – when logic input is non-zero, the connection is made two ½ channels (one input to a BOB and one output port from a BOB). These are configured based on the number of BOBs in the system.

Wave output ports – when logic input is non-zero, a specified wave file is played to a given channel. These are also configured in the Run Time System - .vne file.

Watchdog timers – these objects look for changes in the input and place a signal out for a specified time. These objects are useful for timing events such as how long a sound will play. They are generally used with wave output ports.

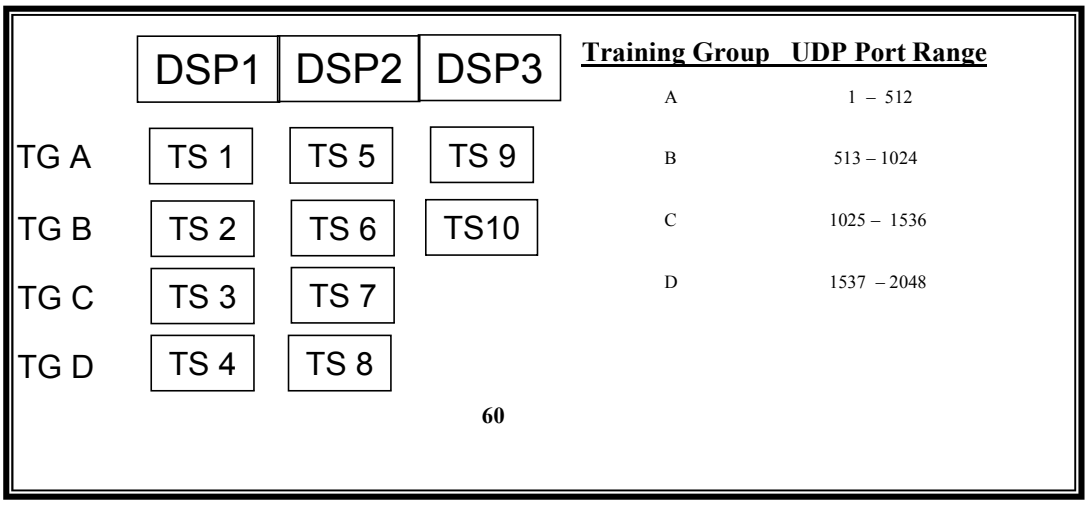
Logic symbols (and, or, not gates, flip flops) – perform normal Boolean operations.

Selectors – used to de-multiplex input coming from the Server in response to VIK key press to generate the proper tone.

Off-page connectors (in and out) – used mostly to keep signal lines from crossing. These objects connect one output symbol to one or more input symbols with the same number. The convention used for the symbol number is generally from which UDP buffer port the data came.

8.5.4.1 UDP/IP Data Buffer definition

The VCS Server communicates to the DSP via a buffer of 32-bit words which is sent over a UDP socket every 70 ms. The diagram below provides an overview of all DSPs in a given main lab. The MINI lab utilizes only the first DSP and the first two training sectors. In addition, input DSP UDP Object 0 is reserved for Master Instructor broadcast and DSP UDP Object 0 is used for heartbeat to the Server. UDP data buffer details are described in Section 8.6.



8.5.4.1.1 Training Sector Buffer Offsets

Each sector in a training group has an offset of 512 from the previous group. Each Training Sector is allocated 512 words which correlate to the UDP port numbers on the des files. Appendix A contains 4 spreadsheets – one for each training group.

8.5.4.1.2 Buffer Offsets within a Training Sector

IATS VCS has allocated 16 channels (2 BOBs) to each training sector. The UDP buffer allows a 16X16 connection matrix in its first 256 entries. Each channel of every BOB is allocated 16 entries even though many of these entries are not used. There are 16 additional rows allocated to other DSP functions. These include volume and gain adjustments, call placed ringing, volume change, override, error and dial tones. UDP data buffer details are described in Section 8.6.

8.5.4.2 Student Position Logic

Logic for Student positions (RPOS and DPOS) is grouped so that inputs coming from that position are together. The logic follows the UDP buffer map row 1 in Figure 8-19 below depicts logic for RPOS. DPOS is similar in content.

Examples:

For the R position to be able to communicate over frequency channel 1, Channel 1 input (to the DSP) must be connected to Channel 11 output (UDP Port 11) **and** RPOS PTT / Side-tone must be enabled/pressed (UDP Port 1). These two values together will turn on the Analog connection from the RPOS mic to Freq 1. The COM Server tells the DSP to enable Port 11 and 161 when the RPOS has a frequency selected on frequency channel 1. The Server tells the DSP to enable Port 1 when the RPOS PTT is depressed. Logic in the Freq section of the design sheet (Figure 8-22) cuts off the frequency input to the RPOS headset while his PTT is depressed.

Hot mic is turned on via an Override UDP port in row 20 of the buffer map (Figure 8-27). When one of the associated OR gate objects (green arrows) is turned on by the Server, the resultant output port is turned on and the position hears side-tone. The override tone is subject to a timer and a volume multiplier which may be altered at the MI position. The user may also hear Side Tone by merely pressing PTT **or** by initiating Position Relief.

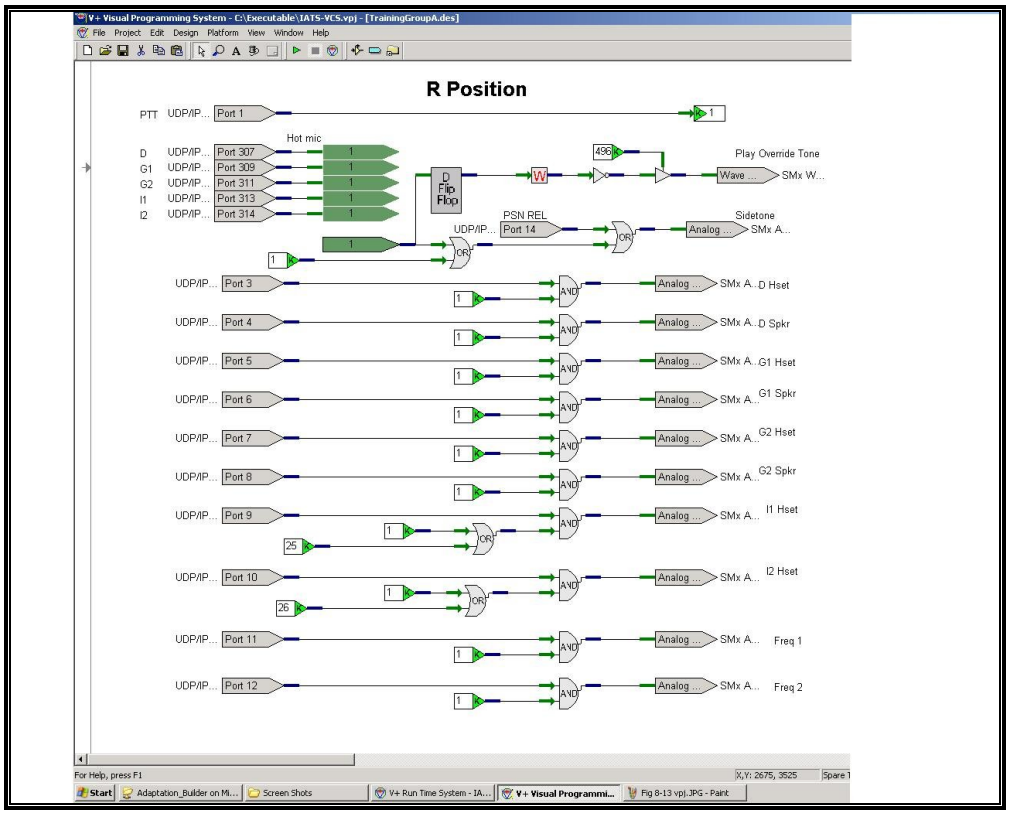


Figure 8-19 V+ Description Sheet – VCS Student (R-Position) Logic

8.5.4.3 Ghost Pilot Position Logic

Ghost Pilot logic is identical to student logic but without Position Relief. Ghost 2 (shown below) is also identical to Ghost 1 with the appropriate UDP buffer ports assigned from the UDP buffer map rows 7 and 5 respectively shown in Figure 8-27**.**

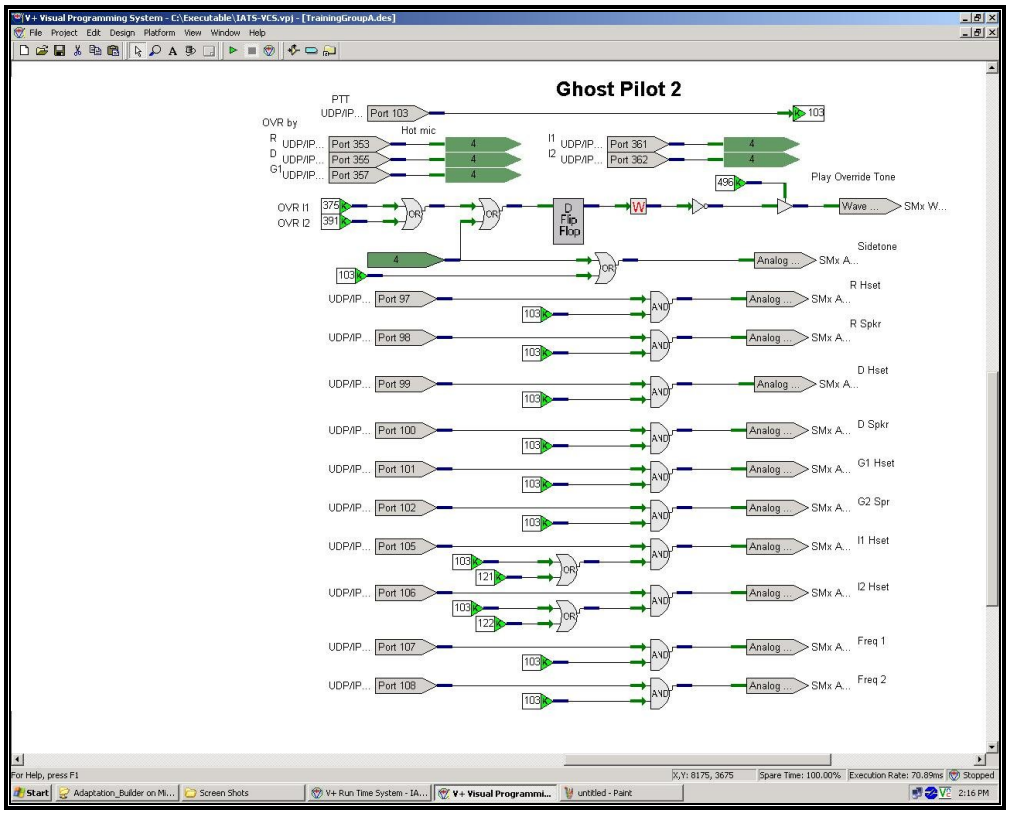
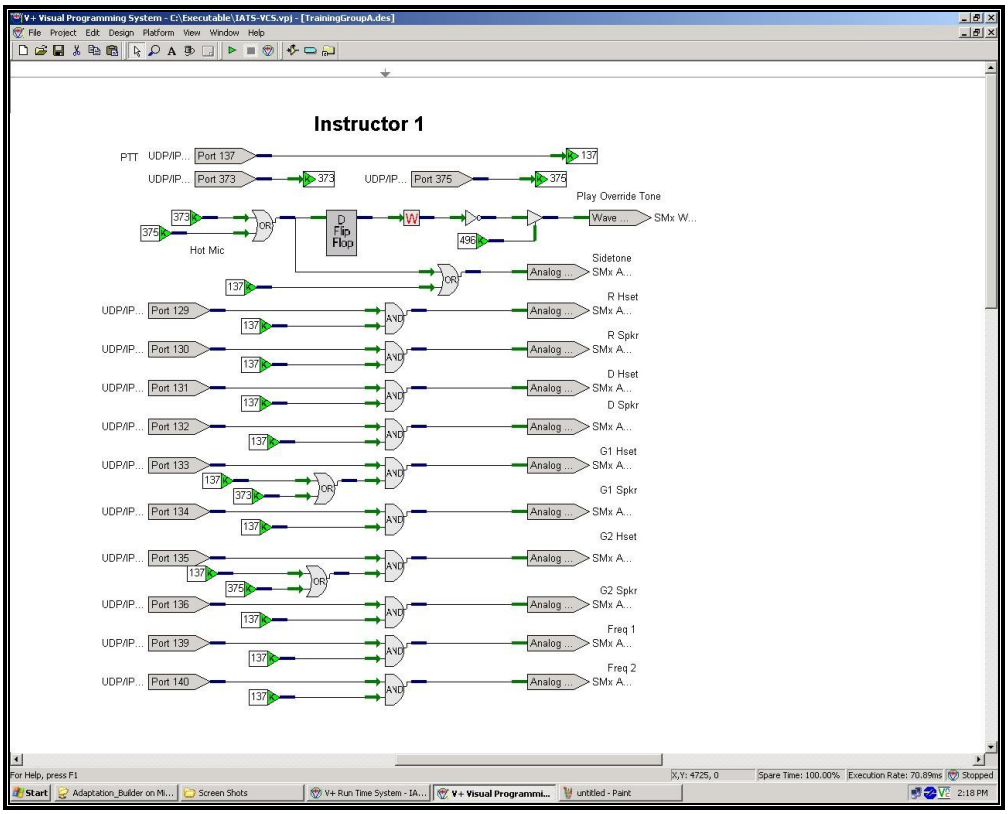
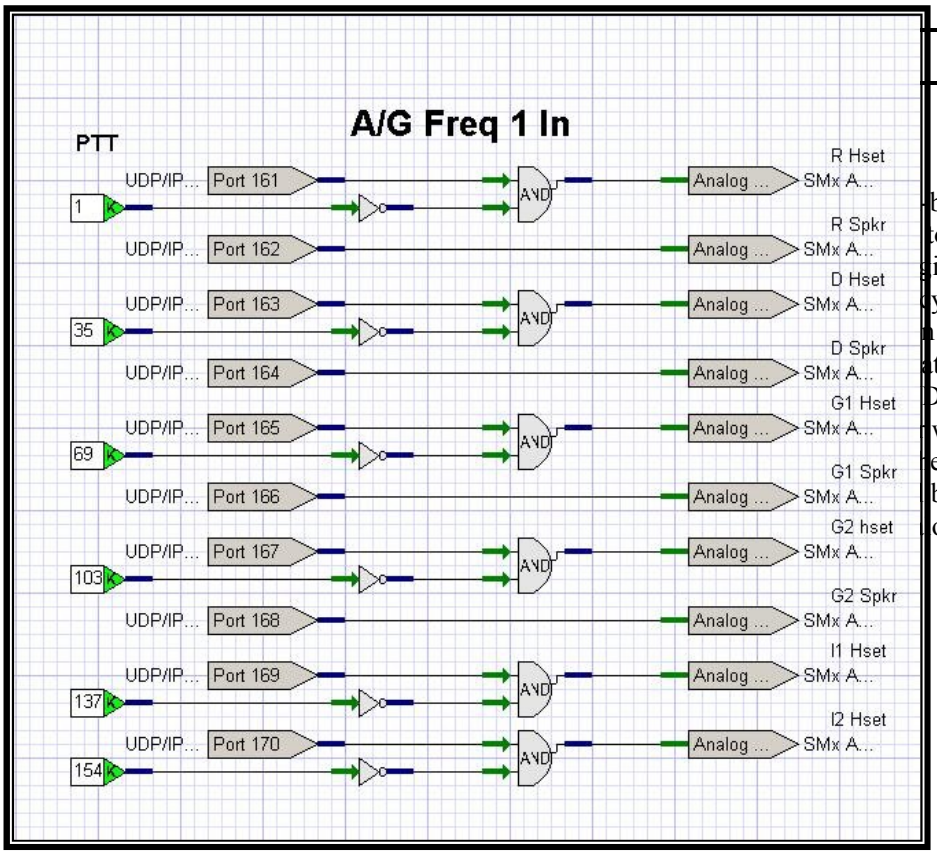


Figure 8-20 V+ Description Sheet – VCS Ghost Pilot (Ghost Pilot 2) Logic

8.5.4.4 Instructor Position Logic

Instructor logic is similar to student except that instructors can only be overridden from a ghost position. Instructor 1 (shown below) is also identical to Instructor 2 with the appropriate UDP buffer ports assigned from the UDP buffer map rows 9 and 10 respectively shown in Figure 8-27.





back.
o the
ic is
y
ata
DP 1
will
e
be 0
og

Figure 8-22 V+ Description Sheet – VCS A/G Frequency Channel (Channel 1) Logic

8.5.4.6 VCS Monitor Logic

The Monitor logic is simple and follows rows 2, 4, 6 & 8 of the UDP buffer map in Figure 8-27. Only students and Ghosts are allowed to be monitored. When an Instructor monitors a student, that UDP input object is also used to turn on the student's hot microphone.

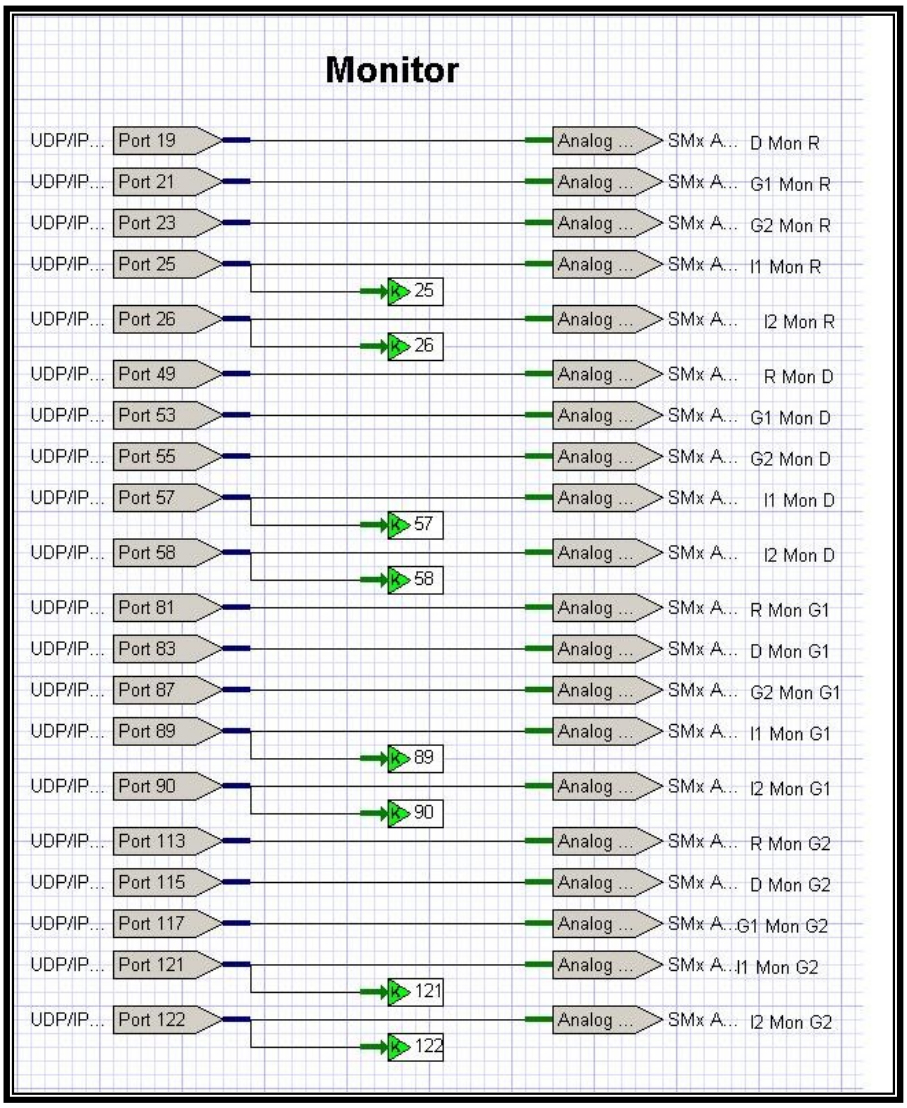


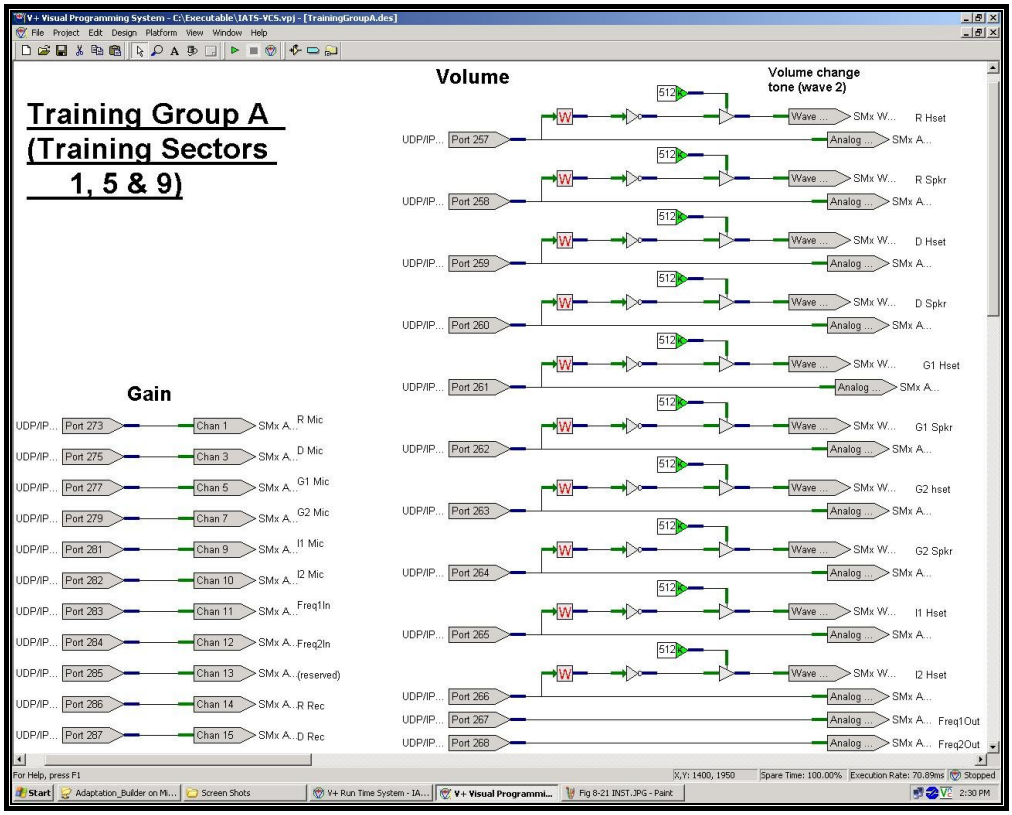
Figure 8-23 V+ Description Sheet – VCS Monitoring Function Logic

8.5.4.7 VCS Volume & Gain Logic

The Volume logic (shown below) allows output levels to headsets and speakers to be altered between 0 and 1.0. The Server initializes all positions to .5 and allows changes to be made from the Client using the Utility Screen (Figure 8-11) in increments of 0.1. The inputs modify row 17 of the UDP buffer map in Figure 8-27. When a volume change is requested, the Volume Change Tone is also played for a short time (via a watchdog timer)

at the new volume setting. This tone is also subject to a volume change multiplier set at the MI position for the entire training sector. Use the middle slider for the desired training sector as shown in Figure 8-2.

The Gain logic (not shown) allows input levels from all microphones to be altered between 0 and 20. The inputs modify row 18 of the UDP buffer map in Figure 8-27. The COM Server initializes all positions to 10 and allows changes in increments of 1. The changes are made on the COM Server (see Figure 8-2) using the on the sliders at the bottom of the screen.



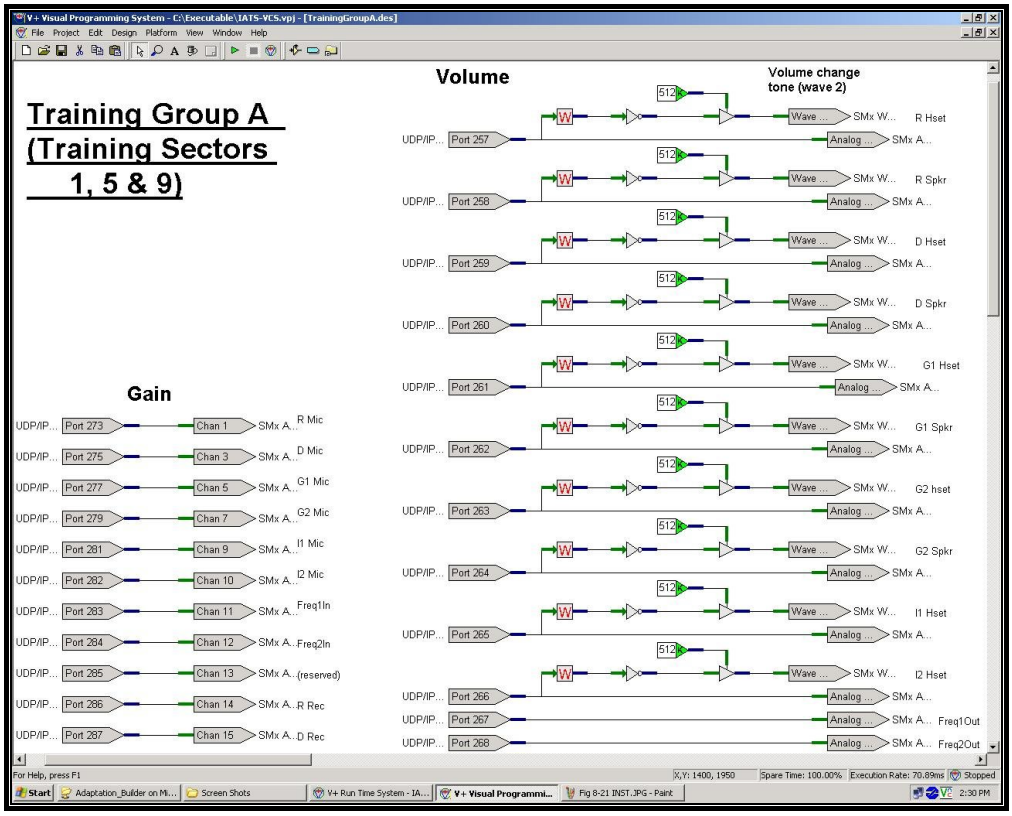


Figure 8-23 V+ Description Sheet – VCS Volume Function Logic

8.5.4.8 VCS Tone Logic

In addition to the volume change, several tones are played in response to Client actions. They are the VSCS audio (.wav) files for Ring, Error and Dial tones. These tones are played via a wave file object (noted in the title) shown on the Common.des description sheet in Figure 8-25.

Ring tone logic (shown below) is straight forward feeding every headset and loudspeaker for all clients. Row 19 of the UDP buffer map, shown in Figure 8-27, contains the port IDs for Ring Tone.

Error tone logic (Figure 8-24) is straight forward - feeding every headset and loudspeaker for all clients. It is subject to a watchdog timer for changes. Row 31 of the UDP buffer map, shown in Figure 8-27, contains the port IDs for Ring Tone.

Dial tone logic (not shown below) is also straight forward - feeding every headset for all clients. Row 30 of the UDP buffer map, shown in Figure 8-27, contains the port IDs for Dial Tone. The Dial Tone is on wave file #5.

The volume of these tones is regulated per TS by manipulating a UDP port (eg. 496 for Override & Error tone volume in Training Group A). The value of the port is adjustable at the COM Server GUI and may be a whole number between 0 and 10.

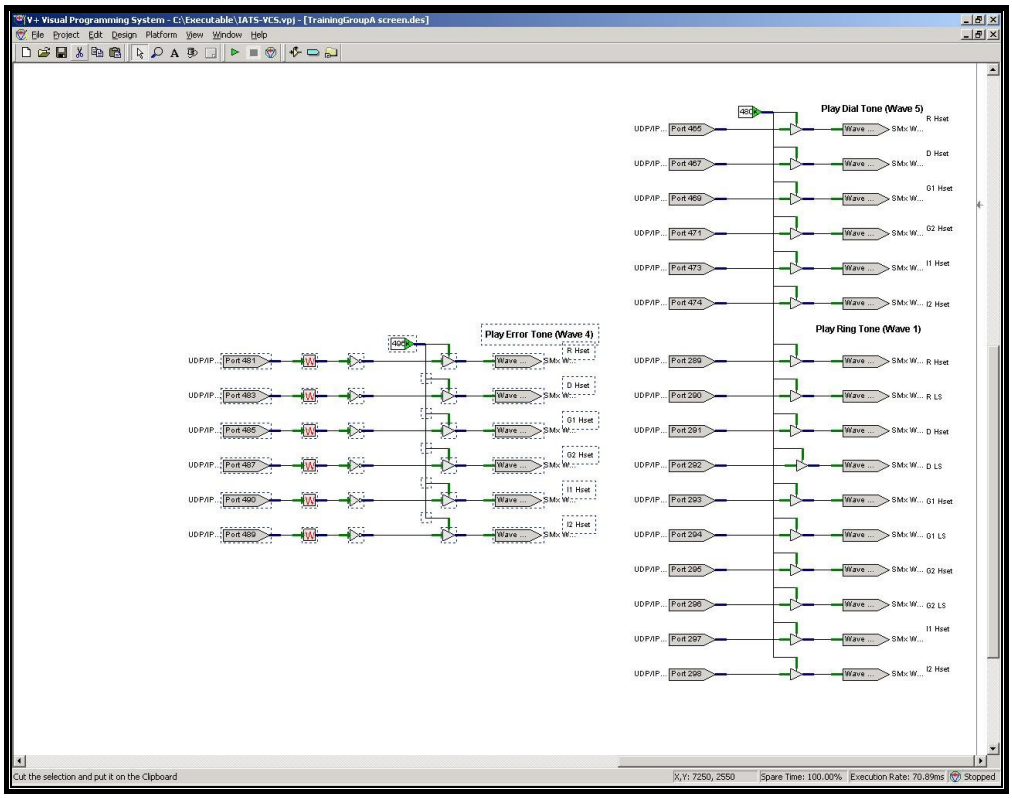


Figure 8-24 V+ Description Sheet – VCS Ring Tone/Error Tone Logic

8.5.4.9 Common VCS Logic

The logic on the other two descriptor sheets is common to all training sectors.

Common.des (Figure 8-25) contains Master Instructor broadcast, wave file player, and heartbeat signal logic. The Master Instructor logic allows one-way broadcast from the MI to all position headsets using UDP Input Port 0.

There are 5 wave file players that continuously play their respective tones. Based on signals from logic in the Training Group sheet the wave file ports are connected to the analog port for the proper speaker/headset. The volume for Dial & Ring are handled separately from Override and Error tones with UDP ports 480 and 496 respectively.

The DSP heartbeat signal is sent to the COM Server over the only UDP output port used (Port 0). The COM Server application indicates when one of the three DSPs (or the only DSP in the mini lab) is down.

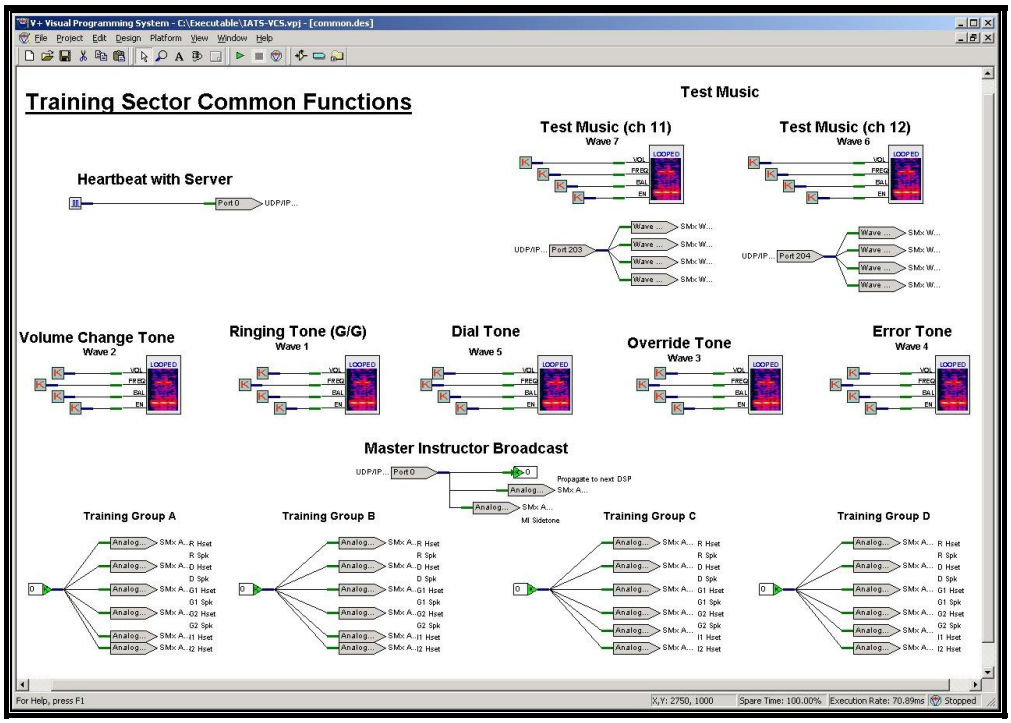


Figure 8-25 V+ Description Sheet – Training Sector Common Functions

VIK Tones.des (Figure 8-26) contains twelve wave file players, one for each VIK key. There is a pair of selectors (R & D) for each training sector in all training groups. There will be a total of 8 selector objects for playing the correct tone while pressing keys on the software VIK. The volume of these tones is regulated per TS by manipulating a UDP port (eg. 512 for Training Group A). The value of the port is adjustable at the COM Server GUI and may be a whole number between 0 and 10. Only Students positions will use the VIK.

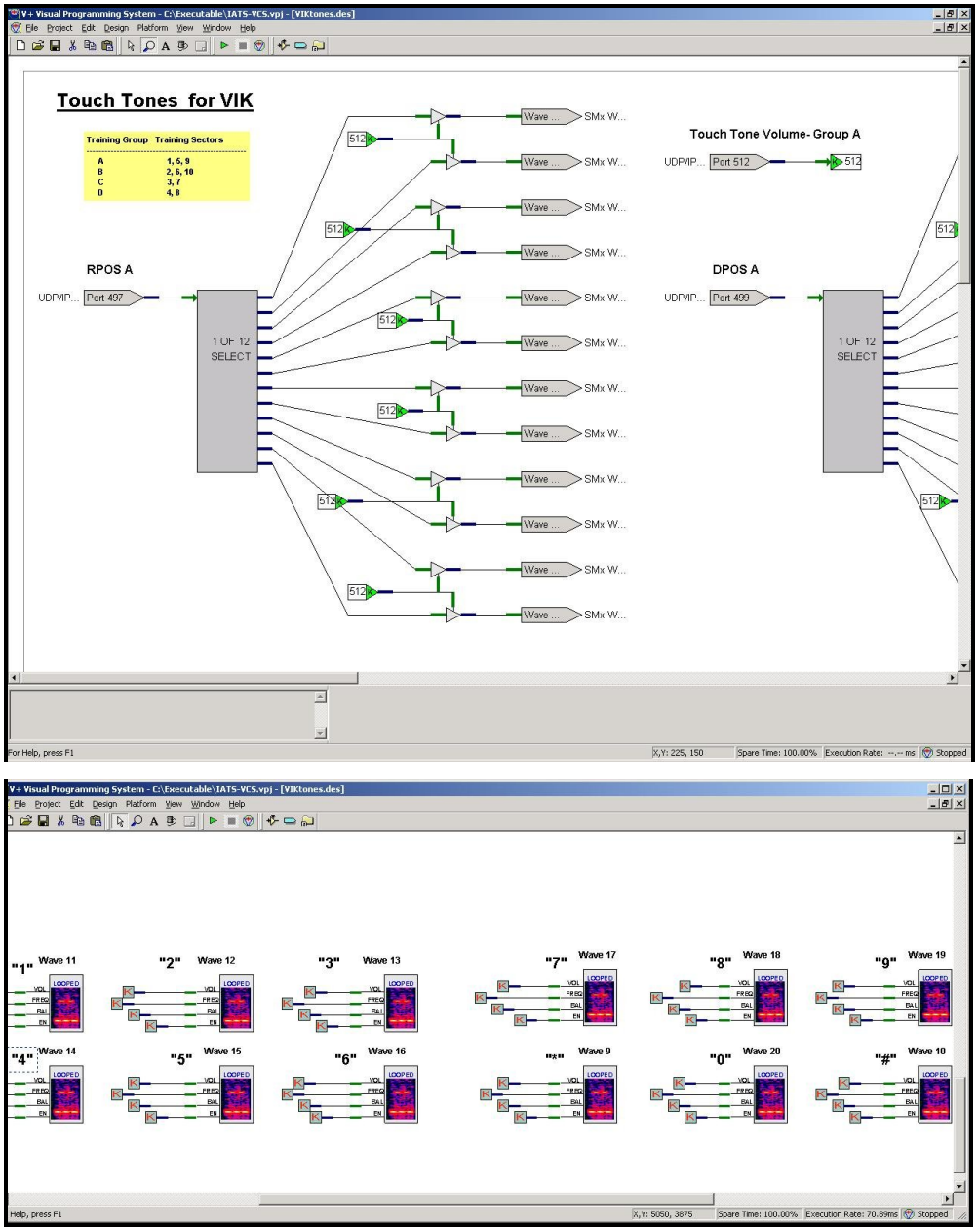


Figure 8-26 V+ Description Sheet – Training Sector Touch Tones for VIK Keys

8.6 VCS COM Server/VCS DSP Interface Control Documentation

8.6.1 Interface Description

This section describes the data conveyed between the IATS Server and DSPs. In each of the main labs at the Academy, the maximum number of clients that will be adapted is sixty (60). A single DSP may control the audio routing of up to 24 clients, six in each of four Training Sectors (TSs). Therefore at least 3 DSPs are required to control the audio routing of 60 clients. A single server application may communicate with up to 3 DSPs simultaneously and therefore control the audio routing of the 60 adapted clients simultaneously. The two-way mechanism for communicating between the DSPs and the server is via a Universal Datagram Protocol (UDP) data burst.

8.6.2 COM Server to DSP Communications

Every adapted interval, the COM Server will look for up to 3 connected DPSs. If it finds any, the data buffer allocated and associated with each DSP is sent to that particular DSP. Regardless of connection status of each DSP, the buffers associated with each DSP are updated as client requests are made that affect the audio routing controlled by that DSP. As a result, as individual clients connect to and register with the server, they become aware of current IC-call and IP-trunk states.

The size of the server's buffer (and therefore the UDP message size) is driven by the number of UDP objects that each DSP allocates to allow it to satisfy the audio routing requirements of the IATS system. In the VCS, each of four training sectors (comprised of six clients) requires a range of 512 UDP objects. Therefore, four training sectors require at least 2048 UDP objects. Each lab also contains a Master Instructor communication UDP object which brings the total number of UDP objects per DSP to 2049. The same port (Port 0) is used in each DSP in the lab and is propagated by the server to all DSPs in a main lab.

Since the DSP requires 32 bits of information per UDP object, the amount of data sent to each DSP every adapted interval is 65568 bits or 8196 bytes. The data that represents the majority of the UDP objects contains the Boolean values 1 or 0. Others objects, especially those that control Gain and Volume contain "real" values that are limited to a defined range. A UDP Port memory map for Training Group A (Training Sectors 1, 5 or 9) is presented in Figure 8-27.

UDP Port Buffer Allocation - Training Group A

Master Inst Broadcast Port 0
TS Offset 0

Training Group Channel and DSP Port Breakout

		Output															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Input	from \ to Ch	R Hset	R spkr	D Hset	D spkr	G1 hset	G1 spkr	G2 hset	G2 spkr	I1 hset	I2 hset	Frq1out	Frq2out	<none>	R rec	D rec	Master
1	Rpos Mic	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	Rpos Monitor	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
3	Dpos Mic	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
4	Dpos Monitor	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
5	G1 Mic	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
6	G1 Monitor	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
7	G2 Mic	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
8	G2 Monitor	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
9	Inst1 Mic	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
10	Inst2 Mic	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
11	AG Freq1 in	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
12	AG Freq2 in	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
13	AudioTest	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
14	R play	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
15	D play	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
16	Master	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256
17	Volume	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272
18	Gain	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288
19	Ring	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304
		R Hset		D Hset		G1 hset		G2 hset		I1 hset	I2 hset						
20	Override R	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320
21	Override D	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336
22	Override G1	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352
23	Override G2	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368
24	Override I1	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384
25	Override I2	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400
																	Volume
30	Dial Tone	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480
31	Error Tone	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496
32	VikTones	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512

- sidetone

Figure 8-27 COM Server – DSP UDP Port Buffer Allocation

8.6.3 DSP to COM Server Communications

Every adapted interval, the DSP will send a 32 bit "heartbeat" message to the Server. The server uses this information to determine if the DSP is still connected. When the number of successive unsuccessful attempts to read from the port used to connect the server to the DSP reaches an adapted value, the Server will mark the DSP down and will discontinue sending data buffers to the DSP until the server can successfully read from that port again. The value of the 32 bit data buffer is not significant, only that data is received indicating that a connection still exists.

8.7 VCS Client/VCS COM Server Interface Control Documentation

8.7.1 Interface Description

This section describes the messages conveyed between the VCS COM Server and VCS clients. The following subsections separate the messages into functionality. The first parameter of each message is the message type and dictates how the message is parsed. Subsequent parameters are separated by the alphanumeric sequence **"##"**. Within each parameter there may be a ***"keyword:value"*** construct or a single or multiple value-only (i.e. no keyword) construct such as ***"value"*** or ***"value:value:value"*** using the ":" alphanumeric for separation. All messages end in the alphanumeric sequence **"!!"**

NOTE - In following sections, the **[|]** indicate parameter options. For example, "**[a | c]"** indicate that "a" or "c" may be entered within a parameter

8.7.2 VCS Client Messages to the VCS COM Server

All of the client-to-server messages are generated as a result of user interaction at the client position. This subsection describes the syntax and format of client-to-server messages.

8.7.2.1 Client Registration

8.7.2.1.1 Client Registration Message Detail

Format:

REGISTER##*Hostname*!!

The REGISTER message is generated at the client position following a client's connect request and subsequent connect accept from the server. The REGISTER message has 1 additional parameter. The Hostname parameter is used to convey the hostname of the registering client so that the server may validate it. Upon successful registration, the COM Server returns to the client the adaptation data that has been loaded for the client. The adaptation data is based on the client hostname (see Section 8.6.3.1).

8.7.2.2 G/G Call Related Messages

8.7.2.2.1 Client Originate Call Message

Format:

GG_ORIGINATE_CALL##*SrcID*##*DestID*##*CallType*##*TrunkID*!!

The GG_ORIGINATE _CALL message is generated at a client position when that client's user wants to call another client within the training sector. The GG_ORIGINATE_CALL message has 3 or 4 additional parameters. The SrcID

parameter is used to convey the source of this message. The DestID parameter is used to convey the destination(s) of this message. If more than one destination is present in the DestID parameter, the individual DestIDs are separated by ":" within the parameter. The CallType parameter can be one of "IC", "MON", "OvrLatch", "OvrNonLatch", "IPHoller", or "IPDirect" and conveys the type of Direct Access (DA) button selected at the source client position. In the case of an InterPhone (IP) selection, the TrunkID parameter conveys the value of the selected trunk as adapted for that DA button, otherwise the TrunkID parameter and the preceding "##" are not part of this message. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio routing is performed.

8.7.2.2.2 Client Answer Call Message

Format:

GG_ANSWER_CALL*##SrcID##DestID##CallType##TrunkID*!!

The GG_ANSWER_CALL message is generated at a client position when that client's user wants to answer an incoming call from another client within the training sector. The GG_ANSWER_CALL message has 3 or 4 additional parameters. The SrcID parameter is used to convey the source of this message. The DestID parameter is used to convey the destination of this message, which is naturally the source of the ORIGINATE_CALL message. The CallType parameter can be one of "IC", "MON", "OvrLatch", "OvrNonLatch", "IPHoller", or "IPDirect" and conveys the type of Direct Access (DA) button selected at the source client position. In the case of an InterPhone (IP) selection, the TrunkID parameter conveys the value of the selected trunk as adapted for that DA button, otherwise the TrunkID parameter and the preceding "##" are not part of this message. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio routing is performed.

8.7.2.2.3 Client/Server Release Call Message

Format:

GG_RELEASE_CALL*##SrcID##DestID##CallType##TrunkID*!!

The GG_RELEASE _CALL message is generated at a client position when that client's user wants to release the current call or place another call to another client within the training sector. The GG_RELEASE _CALL message has 3 or 4 additional parameters. The SrcID parameter is used to convey the source of this message. The DestID parameter is used to convey the destination(s) of this message. If more than one destination is present in the DestID parameter, the individual DestIDs are separated by ":" within the parameter. The CallType parameter can be one of "IC", "MON", "OvrLatch",

"OvrNonLatch", "IPHoller", or "IPDirect" and conveys the type of Direct Access (DA) button selected at the source client position. Additionally, for this message type only, the CallType can also be "RB" indicating that this is the release of a call that has not been answered; it is still in RingBack state. The server this message is sent to performs slightly different logic when releasing an incomplete call vs. a completed call. In the case of an InterPhone (IP) selection, the TrunkID parameter conveys the value of the selected trunk as adapted for that DA button, otherwise the TrunkID parameter and the preceding "##" are not part of this message. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio routing is performed.

8.7.2.2.4 Client Voice Monitor Call Message

Format:

GG_VOICE_MONITOR##SrcID##DestID##ON|OFF!!

The GG_VOICE_MONITOR message is generated at a client position when that client's user wants to monitor what is heard in the headset of another client within the training sector. The GG_VOICE_MONITOR message has 3 additional parameters. The SrcID parameter is used to convey the source of this message. The DestID parameter is used to convey the destination of this message. The ON|OFF parameter can be one of "ON" or "OFF" to indicate that monitoring should be turned on or off, respectively. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio routing is performed.

8.7.2.2.5 Client Join IP Call Message

Format:

GG_JOIN##SrcID##CallType##TrunkID!!

The GG_JOIN message is generated at a client position when that client's user wants to join a trunk call already in progress. The GG_JOIN message has 3 additional parameters. The SrcID parameter is used to convey the source of this message. The CallType parameter can be one of "IPHoller", or "IPDirect" and conveys the type of Direct Access (DA) button selected at the source client position. The TrunkID parameter is used to convey the identity of the trunk that is to be joined. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio routing is performed.

8.7.2.2.6 Client Leave IP Call Message

Format:

GG_LEAVE##SrcID##CallType##TrunkID!!

The GG_LEAVE message is generated at a client position when that client's user wants to leave a trunk call in which they are a participant. The GG_LEAVE message has 3 additional parameters. The SrcID parameter is used to convey the source of this message. The CallType parameter can be one of "IPHoller", or "IPDirect" and conveys the type of Direct Access (DA) button selected at the source client position. The TrunkID parameter

is used to convey the identity of the trunk that is to be left. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio routing is performed.

8.7.2.3 Client AG Frequency Destination Preference Messages

8.7.2.3.1 Client Switch Frequency Channel Routing Message

Format:

AG_AUDIO##HS|LS##FreqID - [1|2]!!

The AG_AUDIO message is generated at a client position when that client's user selects the HeadSet/LoudSpeaker toggle button for a particular frequency on the AG screen. The AG_AUDIO message has 2 additional parameters. The HS|LS parameter is used to convey the selected destination preference. The FreqID parameter is the value "1" or "2" and indicates which of the two frequencies will be directed to the preferred destination. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio routing is performed.

8.7.2.3.2 Client A/G Audio Radio Transmission

Format:

AG_RT##ON|OFF!!

The AG_RT message is generated at a client position when that client's user selects the Radio Transfer (RT) toggle button on the AG screen. The RT button forces all frequency output to the loudspeaker when it is turned on. When the RT button is turned off, frequency output is directed to the setting last selected by the AG_AUDIO message. The AG_RT message has 1 additional parameter. The ON|OFF parameter is used to convey the selected destination preference. The ON|OFF parameter is the value "ON" or "OFF" and indicates loudspeaker output when RT is turned on and AG_AUDIO setting when RT is turned off. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio routing is performed.

8.7.2.4 Client PTT Transmit Indication Messages

8.7.2.4.1 Client Push To Talk On/Off Message

Format:

AG_PTT##ON|OFF!!

The AG_PTT message is generated at a client position when that client's user selects the non-latching software PTT button. The AG_PTT message has 1 additional parameter that conveys "ON" or "OFF to indicate whether the button is pressed or released, respectively. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio routing is performed.

8.7.2.5 Client A/G Frequency Selection Indication Messages

8.7.2.5.1 Client A/G Frequency State Change Message

Format:

AG_SET_STATE##FreqID - [1|2]##XMT|RCV##ON|OFF!!

The AG_SET_STATE message is generated at a client position when that client's user selects a frequency button on the AG screen. Upon selection of the frequency button, the client actually sends two of these messages. The first is sent to turn on the transmit (XMT) half of the frequency button and the second is sent to turn on the receiver (RCV) half of the frequency button. From this state, the XMT and/or RCV halves of the frequency button may be turned on or off separately. The AG_SET_STATE message has 3 additional parameters that convey which frequency is being affected, which portion of the frequency is being affected, XMT or RCV", and how is this frequency portion being affected, ON or OFF. Upon receipt of this message, the server communicates with the DSP so that appropriate audio routing is performed.

8.7.2.6 Client Volume Change Messages for HeadSet and LoudSpeaker

8.7.2.6.1 Client Headset Volume Change Message

Format:

VOLUME_CHANGE_HS##Lower|Higher - [0|1]!!

The VOLUME_CHANGE_HS message is generated at a client position when that client's user selects the headset volume change button on the Utilities screen. Up arrows and down arrows are selected until the volume is at the desired level. When the UP arrow is selected, this message is sent with the Lower|Higher parameter set to 1 indicating an increase in volume, until a maximum is reached. When the DOWN arrow is selected, this message is sent with the Lower|Higher parameter set to 0 indicating an decrease in volume, until a minimum is reached. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio routing is performed.

8.7.2.6.2 Client Loud Speaker Volume Change Message

Format:

VOLUME_CHANGE_LS##Lower|Higher - [0|1]!!

The VOLUME_CHANGE_LS message is generated at a client position when that client's user selects the loudspeaker volume change button on the Utilities screen. Up arrows and down arrows are selected until the volume is at the desired level. When the UP arrow is selected, this message is sent with the Lower|Higher parameter set to 1 indicating an increase in volume, until a maximum is reached. When the DOWN arrow is selected, this message is sent with the Lower|Higher parameter set to 0 indicating an decrease in

volume, until a minimum is reached. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio routing is performed.

8.7.2.7 VIK Dialing Indication messages

8.7.2.7.1 Client VIK Key Press Indication Message

Format:

KEYPRESSED##KeyNumber - [0-11]##ON|OFF

The KEYPRESSED message is generated at a client position when that client's user places a call via the Very Important Kommunication (VIK) software device. The KeyNumber parameter will convey the value of the key VIK and the ON|OFF parameter indicates whether the key is pressed or released. Upon receipt of this message, the server communicates with the DSP to so that appropriate audio tone is heard at the client position.

8.7.3 VCS COM Server to VCS Client Messages

The majority of server-to-client messages are generated in response to a client action, however the 'SHUTDOWN_CLIENT' messages are used when the Master Instructor manually shuts down a particular client or group of clients. This subsection describes the syntax and format of server-to-client messages. Functional use of these messages is described in Section 8.6.1.

8.7.3.1 Adaptation Messages

Adaptation messages are used to convey specific adaptation information to a specific registering client. Upon receipt of a connect request from a particular Training Sector (TS) client, the server will determine the appropriate adaptation set information to deliver to the client, based on the client's hostname. The set of adaptation information consists of at least 1 POS_ID message, 1 AG_SCREEN message and 1 GG_SREEN message. If a particular client has more than 1 AG or GG screen adapted then a second AG_SCREEN or GG_SCREEN message will be delivered as well. The messages are built using the training sector adaptation text file data described in Section 9.2.

8.7.3.1.1 Server to Client Position ID Message

Format:

POS_ID##PosID:xxx##PosType:yyy!!

The POS_ID message has 2 additional parameters. The additional parameters are in a "keyword:value" format. The PosID parameter conveys the default position identifier used by the client, for example, "PosID:R66". The PosType parameter conveys the position type used by the client, such as "PosType:ATC" for student positions or "PosType:PILOT" for ghost pilot positions (see Section 9.2.1.3).

8.7.3.1.2 Server to Client A/G [1/2] Screen Adaptation Message

Format:

AG_SCREEN1##AG1 Information | AG_SCREEN2##AG2 Information!!

The AG_SCREENx message has many additional parameters. The additional parameters are all in a "keyword:value" format. The AG_SCREENx message conveys the information necessary to configure the AG1 and AG2 screens at the client position.

8.7.3.1.3 Server to Client G/G [1/2] Screen Adaptation Message

Format:

GG_SCREEN1##GG_SCREEN1 Information!!

The GG_SCREENx message has many additional parameters. The additional parameters are all in a "value" format. The GG_SCREENx message conveys the information necessary to configure the GG1 and GG2 screens at the client position.

8.7.3.2 GG_Call Related Messages

8.7.3.2.1 Server to Client G/G incoming Call Message

Format:

GG_INCOMING_CALL##SrcID##DestID##CallType##TrunkID!!

The GG_INCOMING_CALL message is generated at the server upon receipt of a GG_ORIGINATE_CALL message from a client. The GG_INCOMING_CALL message has 3 or 4 additional parameters. The SrcID parameter is used to convey the source of the incoming call. The DestID parameter is used to convey the single destination of the incoming call. The CallType parameter can be one of "IC", "MON", "OvrLatch", "OvrNonLatch", "IPHoller", or "IPDirect" and conveys the type of Direct Access (DA) button selected at the source client position.

In the case of an InterPhone (IP) selection, the TrunkID parameter conveys the value of the selected trunk as adapted for that DA button, otherwise the TrunkID parameter and the preceding "##" are not part of this message.

8.7.3.2.2 Server to Client G/G Call Answered Message

Format:

GG_CALL_ANSWERED##SrcID##DestID##CallType##TrunkID!!

The GG_CALL_ANSWERED message is generated at the server upon receipt of a GG_ANSWER_CALL message from a client. The GG_CALL_ANSWERED message has 3 or 4 additional parameters. The SrcID parameter is used to convey the source of the answering client. The DestID parameter is used to convey the single destination for this message. The CallType parameter can be one of "IC", "MON", "OvrLatch",

"OvrNonLatch", "IPHoller", or "IPDirect" and conveys the type of Direct Access (DA) button selected at the source client position.

In the case of an InterPhone (IP) selection, the TrunkID parameter conveys the value of the selected trunk as adapted for that DA button, otherwise the TrunkID parameter and the preceding "##" are not part of this message.

8.7.3.2.3 Server to Client Call Released Message

Format:

GG_CALL_RELEASED##SrcID##DestID##CallType##TrunkID!!

The GG_CALL_RELEASED message is generated at the server upon receipt of a GG_RELEASE_CALL message from a client. The GG_CALL_RELEASED message has 3 or 4 additional parameters. The SrcID parameter is used to convey the source of the releasing client. The DestID parameter is used to convey the single destination for this message. The CallType parameter can be one of "IC", "MON", "OvrLatch", "OvrNonLatch", "IPHoller", "IPDirect", or "Shutdown" and conveys the call type being released. The "Shutdown" call type is only used in the case when the server is informing the client to shutdown the client application after releasing all outstanding calls.

In the case of an InterPhone (IP) selection, the TrunkID parameter conveys the value of the selected trunk as adapted for that DA button, otherwise the TrunkID parameter and the preceding "##" are not part of this message.

8.7.3.2.4 Server to Client Override Initiated Message

Format:

GG_OVERRIDE_INITIATED##SrcID##DestID##OverType!!

The GG_OVERRIDE_INITIATED message is generated at a client position when a client overrides another client position. The GG_OVERRIDE_INITIATED message has 3 additional parameters. The SrcID parameter is used to convey the source of the overriding client. The DestID parameter is used to convey the single destination for this message. The OverType parameter can be one of "OvrLatch", or "OvrNonLatch" and conveys the override type.

8.7.3.3 GG_In-Use Indicator Messages

8.7.3.3.1 Server to Client G/G Call Status On Message

Format:

GG_STATUS_ON##SrcID##DestID##CallType!!

The GG_STATUS_ON message is generated at the server after a GG_CALL_ANSWERED message is processed. It is sent to all clients in the training sector to inform them that a call is in progress within the training sector and a particular DA button should be marked IN-USE, currently depicted as a green bar at the top of the

DA button. The GG_ STATUS_ON message has 3 additional parameters. The SrcID parameter is used to convey one participant in the call and the DestID parameter is used to convey the other participant in the call. The CallType parameter can be one of "IC", "MON", "OvrLatch", or "OvrNonLatch" and conveys the DA button (call) type to be marked IN-USE.

8.7.3.3.2 Server to Client G/G Call Status on Message

Format:

GG_STATUS_OFF##SrcID##DestID##ButtonType!!

The GG_STATUS_OFF message is generated at the server after a GG_ STATUS_OFF message is processed. It is sent to all clients in the training sector to inform them that a call is no longer in progress within the training sector and a particular DA button should be marked INACTIVE by removing the green bar at the top of the DA button. The GG_STATUS_OFF message has 3 additional parameters. The SrcID parameter is used to convey one participant in the call and the DestID parameter is used to convey the other participant in the call. The ButtonType parameter can be one of "IC", "MON", "OvrLatch", or "OvrNonLatch" and conveys the DA button type to be marked INACTIVE.

8.7.3.4 AG_Frequency_XMTR_In-Use Button Messages

8.7.3.4.1 Server to Client PTT Begin Indication Message

Format:

PTT_BEGIN##FreqID[1|2]!!

The PTT_BEGIN message is generated at the server when the server detects that the PTT device is pressed at a particular client and then it is sent to that client. The PTT_BEGIN message has 1 additional parameter. The FreqID parameter value may contain the value 1 or 2. Only 2 frequencies are permitted per training sector. Either one or both may be adapted as Emergency or Non-Emergency. When the hardware PTT device is pressed at the client position, this message is built for adapted Non-Emergency channels only. When the non-latching software Emergency PTT is pressed at the client position, this message is built for adapted EMERGENCY and NONEMERGENCY channels. Upon receipt of this message, the client's specified frequency's XMTR indicator changes from green to amber.

8.7.3.4.2 Server to Client PTT End Indication Messages

Format:

PTT_END##FreqID[1|2]!!

The PTT_END message is generated at the server when the server detects that the PTT device is released at a particular client and then it is sent to that client. The PTT_END message has 1 additional parameter. The FreqID parameter value may contain the value 1

or 2. Only 2 frequencies are permitted per training sector. Either one or both may be adapted as Emergency or Non-Emergency. When the hardware PTT device is released at the client position, this message is built for adapted Non-Emergency channels only. When the non-latching software Emergency PTT is released at the client position, this message is built for adapted Emergency and Non-Emergency channels. Upon receipt of this message, the client's specified frequency's XMTR indicator changes from amber to green.

8.7.3.5 AG_Frequency_RCVR_In-Use Button Messages

8.7.3.5.1 Server to Client PTT Notify Indication Message

Format:

PTT_NOTIFY##ON|OFF##FreqID - [1|2]!!

The PTT_NOTIFY message is generated at the server when the server detects that a PTT device is pressed at a particular client and then it is sent to all other clients in that training sector. The PTT_NOTIFY message has 2 additional parameters. The ON|OFF parameter value is either "ON or "OFF". The FreqID parameter value may contain the value 1 or 2. Upon receipt of this message, the client's frequency's RCVR indicator should be turned on or off, respectively. Upon receipt of this message with an "ON" value, the client's specified frequency's RCVR indicator changes from green to orange. Upon receipt of this message with an "OFF" value, the client's specified frequency's RCVR indicator changes from amber to green.

8.7.3.6 Client Health Check Messages

8.7.3.6.1 Client Heartbeat Message

Format:

heartbeat##UP|DOWN!!

The heartbeat message is generated at the server once every adapted interval and conveys to the client receiving this message that the connection status of the client's associated DSP is up or down. The heartbeat message has 1 additional parameter with a value of "UP" or "DOWN".

8.7.3.7 Shutdown Messages

8.7.3.7.1 Server to Client Shutdown Message

Format:

SHUTDOWN_CLIENT##Text!!

The SHUTDOWN_CLIENT message is generated at the server when the Master Instructor manually shuts down a client or group of clients or the server detects an invalid registration message from the client. The SHUTDOWN_CLIENT message has 1 additional parameter. The Text parameter value is used to convey the reason the server is

directing the client to shutdown. The Text parameter value is displayed at the client position in a modal popup message box prior to the client shutting down. Three reasons currently used by the server are “Client has been shutdown by server”, “Invalid Host Name Error from Server”, and “Duplicate Host Name Error from Server”.

8.7.3.8 Error Messages

8.7.3.8.1 Server to Client Error Message

Format:

ERROR##SrcId##DestID##Text!!

The ERROR message is generated at the server when the Master Instructor detects that a particular client is trying to communicate with another client in the same training sector and the destination client is not currently connected to the server. The ERROR message has 3 additional parameters. The SrcID parameter is used to identify the source of the message, the DestID parameter is used to identify the destination of the message and the Text parameter is currently set to “Destination not connected to server”. The Text parameter value is displayed at the client position.

9 VCS Adaptation

9.1 VCS COM Server Adaptation

9.1.1 Microsoft Host File

The HOSTS file located in the IATS COM Server at location c:\ Windows\system32\drivers\etc\HOSTS contains the Internet Protocol (IP) address and Hostname (Windows Computer Name) of all the computers in the IATS VCS, including the server, clients, DSPs, and SDE interfaces. All VCS machines are assigned the appropriate IP address and name during the initial software installation from this Hosts file. It is also used to find and validate clients connecting to and registering with the VCS COM Server. A copy of the HOSTS file can be found in Appendix B of this document.

9.1.2 Client Computer Name and its Significance

All Clients under a VCS COM Server control have unique computer names that are specified during software installation. These specified names assist the COM Server in determining how a connection request from a client will be handled. Client hostnames specify lab location (LAB1, LAB2, or MINI), training sector number (TS1-TS10), and position type (RPOS, DPOS, GST1, GST2, INT1, or INT2). LAB1-TS1-RPOS, for example, is the hostname of the RPOS student in training sector 1 in laboratory 1. This information allows the VCS COM Server to determine which audio interfaces correspond

to a connecting client as well as what adaptation data to send to the client after accepting the connection.

9.2 VCS Client Adaptation

A VCS client adaptation is a folder containing a set of six sub-directories – one for each position in the training sector. The directory that holds these six folders – whose name is also that of the adaptation – may also have a validation file created from the Adaptation Builder. The underlying structure has the directories (RPOS, DPOS, GST1, GST2, INT1, INT2) corresponding to the six client positions (See Figure 9-1).

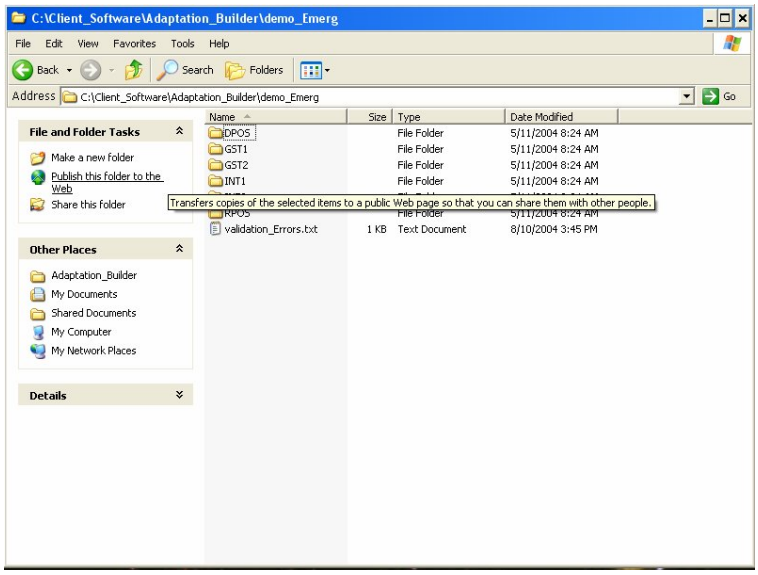


Figure 9-1 Adaptation Set Subfolder Layout

Each position folder contains a set of five files as described section 9.2.1. NOTE – these files are ASCII files, however they should not be modified outside of the VCS Adaptation Builder which is described in Section 9.3. Figure 9-2 shows the position folder file listing.

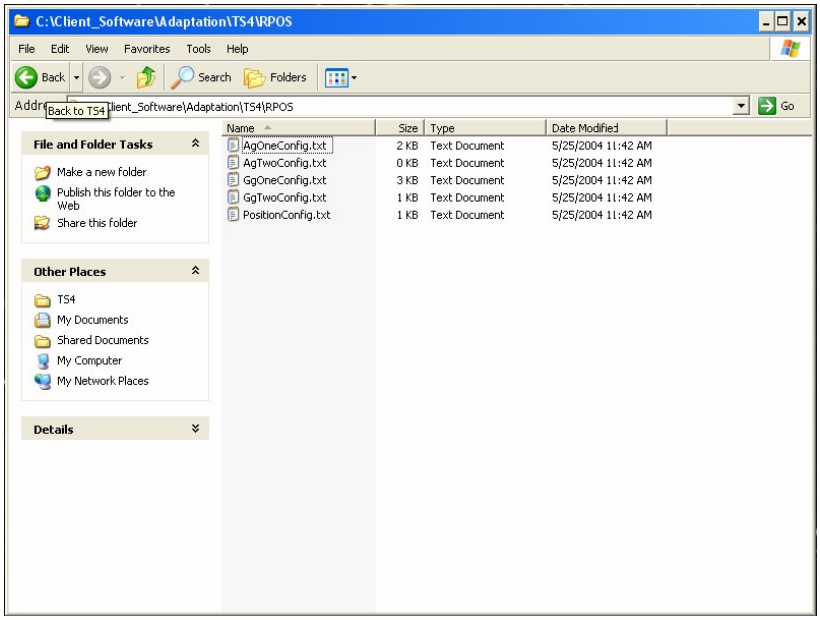


Figure 9-2 Client Position Subfolder Layout

9.2.1 Client Adaptation Design

The VCS Client adaptation consists of five text files which describe the screen buttons and logical position information. There are two ground-to-ground files and two air-to-ground files which correspond to the AG and GG screens on the Client. There is also a Position Configuration file that describes the Clients logical ID and its position type.

The three individual adaptation file types are described below. Comments can be added in any of the files by placing a hash symbol (#) in the first character on the line – the remainder of the line is ignored by the Servers adaptation file parser. If a given screen has no buttons defined, the corresponding file must still exist. Examples of each file type are found in Appendix C.

9.2.1.1 A/G adaptation text files

The two air-to-ground adaptation files (AgOneConfig.txt and AgTwoConfig.txt) have 13 parameters – only 5 of which are modifiable. They follow the form

<Parameter> :< value>

AG file parameters and possible values are described below.

PosId: position ID integer 1 - 12

FrqId: Frequency ID string 1 – 7 characters (eg.132.500)

FrqName: Frequency Name string 1 – 7 characters (eg.GWO)

FrqChannel: Frequency Channel integer 1 – 2 determines which of the two physical channels this frequency is assigned

FrqType: Frequency Type NonEmergency or Emergency When using the Adaptation Builder, this value is auto-assigned so that all frequencies on a particular Frequency Channel are the same type.

The remaining parameters (**RemoteMute**, **BUEC**, **PTTPreempt**, **GroupMaintenance**, **MainStbyXMTR**, **MainStbyRcvr**, **XCouple**, and **GroupControl**) are not used by the Server or the Client and are currently set to "no" by the Adaptation Builder.

9.2.1.2 G/G adaptation text files

The two ground-to-ground adaptation files (GgOneConfig.txt and GgTwoConfig.txt) have up to 25 stanzas – one for each possible DA button. Each stanza has nine lines. Note that GG data is positional verses tagged as in AG and PosID files.

Line	Name	Description	Example
1	Position	button pos identifier. Integer 1-25	1
2	Destination	List of names matching logical posID or SIM ID on another Client	ZFW_48HG1:ZFW_48HG2
3	Label	Up to 5 lines each up to 7 chars. Lines are separated by colons	ZFW:48::367
4	Button type	Latching or NonLatching	Latching
5	Call type	IP, IC, VOICEMON, OVR	IP
6		Holler or NonHoller	Holler
7		Dial or NonDial	NonDial
8	Trunk ID	IP calls only. 2/3 digit integer	48
9	Source ID	Up to 12 char. String, needs to match a Destination string somewhere else in the training sector	RPOS

There also must be a blank line in between stanzas for each button.

9.2.1.3 Position Configuration ID text file

The position configuration file (PositionConfig.txt) is a small file with just two entries

PosID: Logical Position ID string of 3-4 chars. Identifier is shown to other positions during OVR or MON calls etcs.

PosType: Logical Position Type ATC / Pilot / Inst. Currently not used.

9.3 VCS Client Adaptation Builder

9.3.1 Adaptation Builder Application Description

To maintain the proper file formatting within the client adaptation files, the VCS system provides an Adaptation Builder Application for building adaptation sets. The Builder is designed to allow the adaptation Subject Matter Expert (SME) to visualize the client screens as they would be displayed in the client application. It also allows them to add and remove A/G or G/G communication button adaptation to and from the adaptation set. The Adaptation Builder Application reads and writes data using the VCS adaptation data file formatting described in Section 9.2.

Upon application startup a Lab selection dialog is displayed (Figure 9-3). This dialog maps the Adaptation Builder application to the adaptation sets residing on the user-selected lab's COM Server. Each COM Server may contain a different collection of adaptation sets.

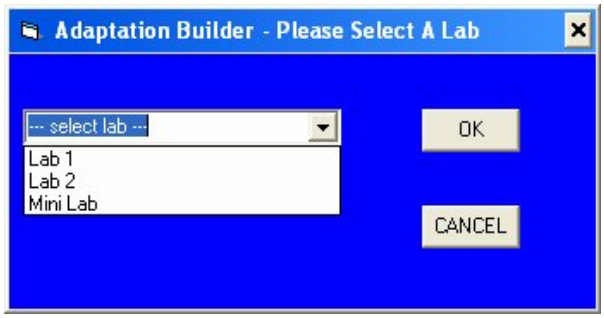


Figure 9-3 Adaptation Builder Lab Selection Screen

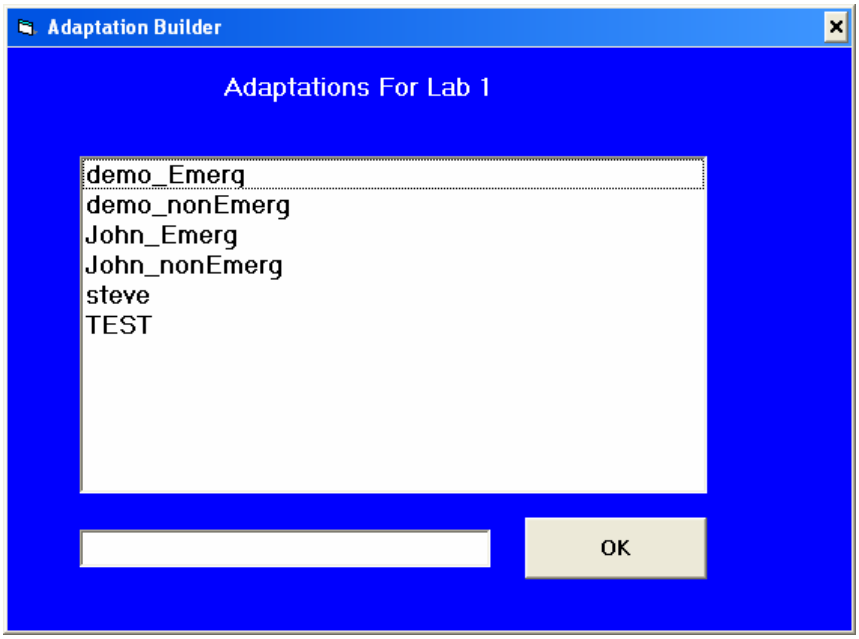


Figure 9-4 Adaptation Builder Adaptation Selection List for Lab 1

Once the Lab is selected, the user is prompted to select an adaptation set (Figure 9-4). The adaptation sets that appear in the selection list are also the same sets that are available for training sector load using the COM Server Adaptation Load process described in Section 8.1.2.5.

The user may also select a new adaptation set by entering a new name in the provided textbox within the selection list dialog.

Upon specifying an adaptation set the Adaptation Builder Main Form is displayed.

9.3.1.1 Adaptation Main Form

The Builder's main form is a two-level multi-tabbed formed. The first tab level contains a tab for each position within the training sector (RPOS, DPOS, INT1, INT2, GST1, and GST2). The second tab (Sub-tab) level contains a tab for each Screen within a position as well as a tab for Position Identification (A/G1, A/G2, G/G1, G/G2, POSID). Each of these Sub-tab screen allows the user to save the adaptation set as the current name or as a new name (using the save as option).

9.3.1.1.1 A/G (1/2) Sub-tab

The A/G Sub-tab, depicted in Figure 9-3, allows the user to define Air/Ground Frequency Button that would appear on that A/G Screen at the position indicated in the main tab. A set of control buttons run along the bottom of the Sub-tab. These buttons allow the user

to save the adaptation set using the current name, save the adaptation set as another name (note – upon selection of "Save As", the user will be prompted for which lab the new adaptation set should be located), or exit the program. A total of twelve frequency button may be defined for each A/G Screen. The buttons are laid out in the Sub-tab the way they will appear on the A/G 1 or A/G 2 Screen at the client upon initialization. The buttons are defined by left-clicking on the button in the desired screen location. A right-click on a frequency button will present a menu that allows the user to COPY / CUT the detail information from that button onto a "clipboard". If a copy or cut operation has already been performed, a PASTE operation is also allowed for the frequency button from the "clipboard".

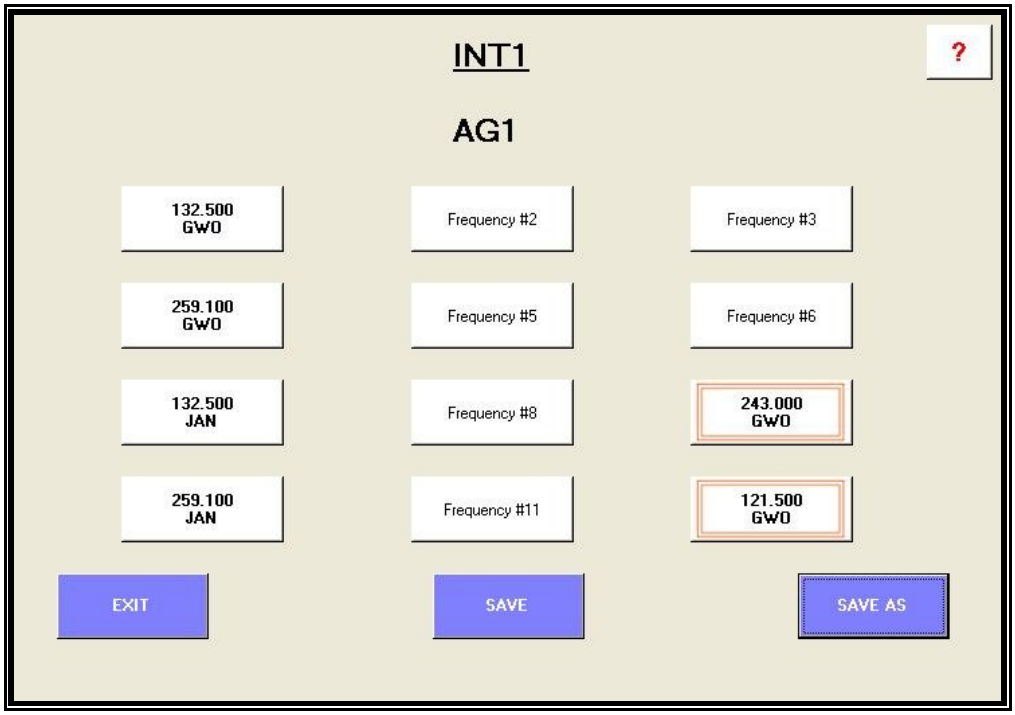


Figure 9-5 Adaptation Builder A/G 1 Screen for Position INT1

Upon the left-clicking Frequency ID button, the A/G Frequency Detail Screen (depicted in Figure 9-6) is displayed. This screen allows the user to define exactly how that particular frequency button will look and operate.

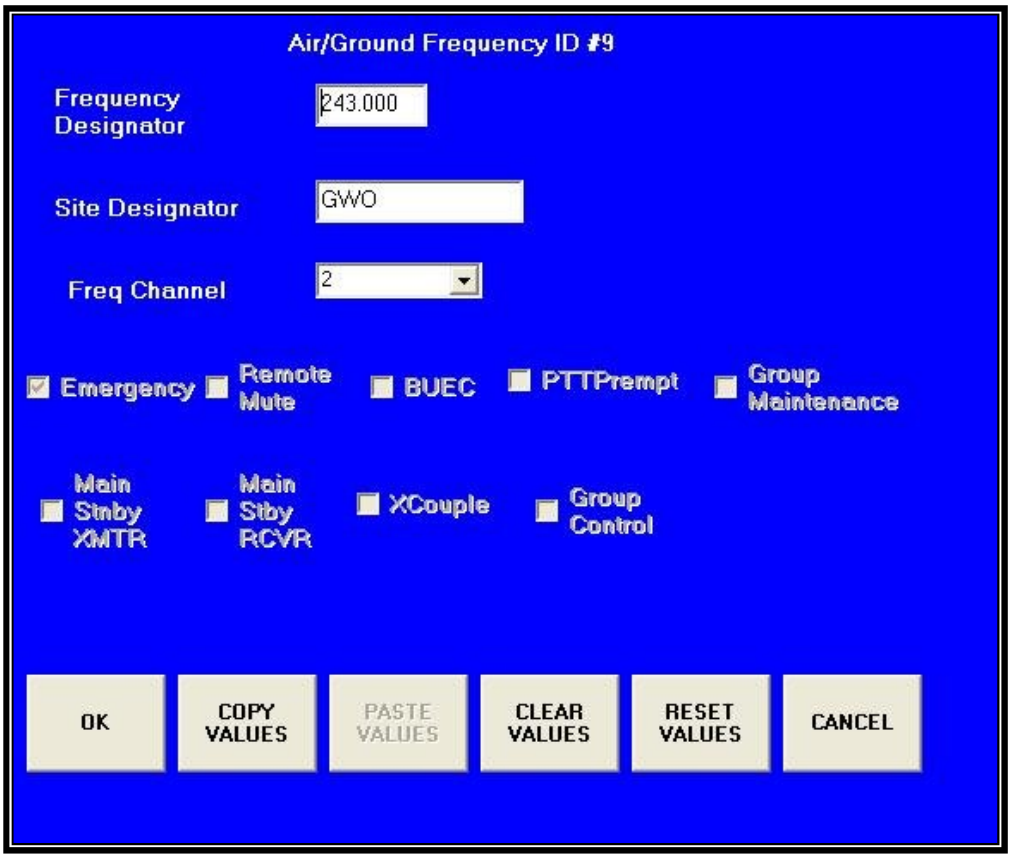


Figure 9-6 Adaptation Builder A/G Frequency Detail Screen

The A/G Frequency Detail Screen provides textboxes to enter the Frequency Designator (number) and Site Designator (Name) which both appear in the frequency button on the client A/G Screen upon initialization. The entries are limited to 7 alphanumeric characters. This screen also provides a pull down menu to designate which frequency channel the frequency button would be assigned to.

NOTE - Because of the IATS VCS architecture, the system is limited to two frequency channels.

The remaining items are checkboxes that are currently read only. The emergency check box is set based on the frequency channel selected for this frequency button and if that frequency channel is configured as an emergency frequency. This setting is configured in the Position ID Detail Screen (described in Section 9.3.1.1.3). The rest of the check

boxes depict other VSCS-style A/G frequency attributes which are provided in this screen for potential future VCS enhancements and are currently not used and unchangeable.

There are also a set of control buttons along the bottom of the screen to allow the user to complete data entry for Frequency ID button (OK), copy all the data values to a "clipboard"-type buffer for pasting them into another AG Frequency Button Detail Screen later, paste a previously copied DA Button screen data set to the "clipboard" buffer. This is the same clipboard used by the right-click menu mentioned above. There is a clear all data values on the screen, reset all data values to their value when the DA Button Detail Screen was displayed, and cancel to return to the GG Sub-tab without saving any changes.

9.3.1.1.2 G/G (1/2) Sub-tab

The G/G sub-tab, depicted in Figure 9-7, allows the user to define G/G Direct Access (DA) call buttons that would appear on a G/G Screen at the position indicated in the main tab. A total of 25 buttons may be adapted per G/G Screen.

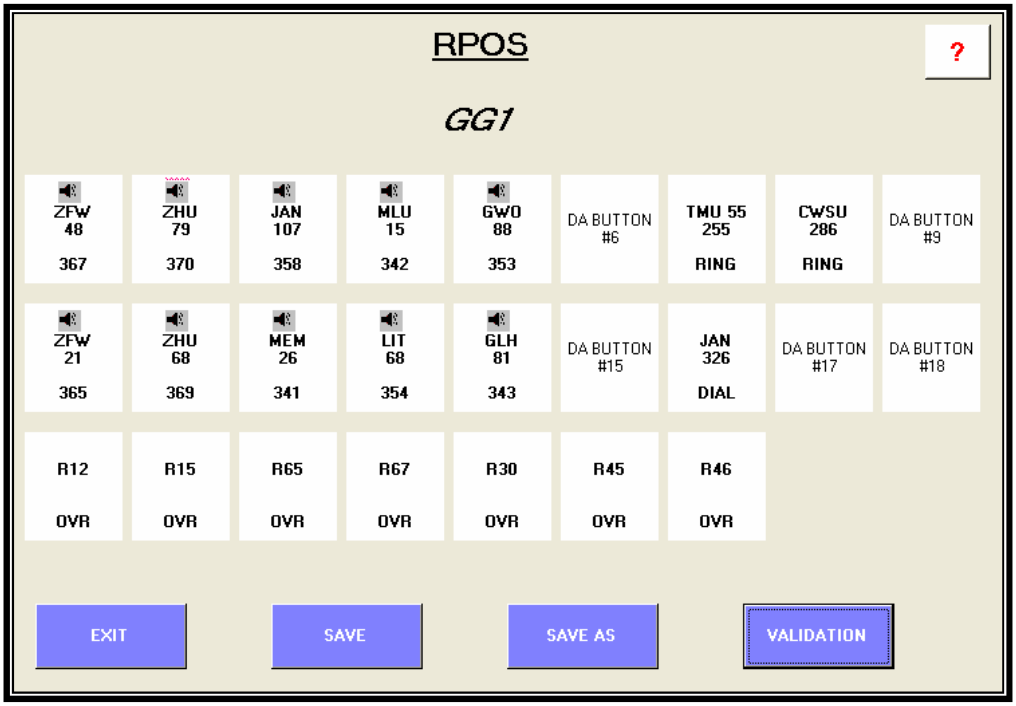


Figure 9-7 Adaptation Builder G/G1 Screen for Position RPOS

The DA buttons are laid out in the sub-tab screen the way they will appear in the specified G/G screen at the VCS client upon initialization. The buttons are defined by clicking on the DA button in the desired screen location. A set of control buttons run along the bottom of the sub-tab. These buttons allow the user to save the adaptation set using the current name, save the adaptation set as another name or exit the program. The VALIDATION button function is described in Section 9.3.2.

A right-click on a frequency button will present a menu that allows the user to COPY / CUT the detail information from that button onto a "clipboard". If a copy or cut operation has already been performed, a PASTE operation is also allowed for the frequency button from the "clipboard".

NOTE - Upon selection of "SAVE AS", the user will be prompted for which lab the new adaptation set should be located),

Upon the left-clicking a DA call button on the G/G sub-tab, the DA Button Detail Screen, depicted in Figure 9-8, is displayed. This screen allows the user to define exactly how a particular DA button will look and operate. There are also a set of control buttons along the bottom of the screen to allow the user to complete data entry for the button (OK), copy all the data values to a "clipboard"-type buffer for pasting them into another DA Button Detail Screen later, paste a previously copied DA Button screen data set to the "clipboard" buffer, clear all data values on the screen, reset all data values to their value when the DA Button Detail Screen was displayed, and cancel to return to the GG Sub-tab without saving any changes.

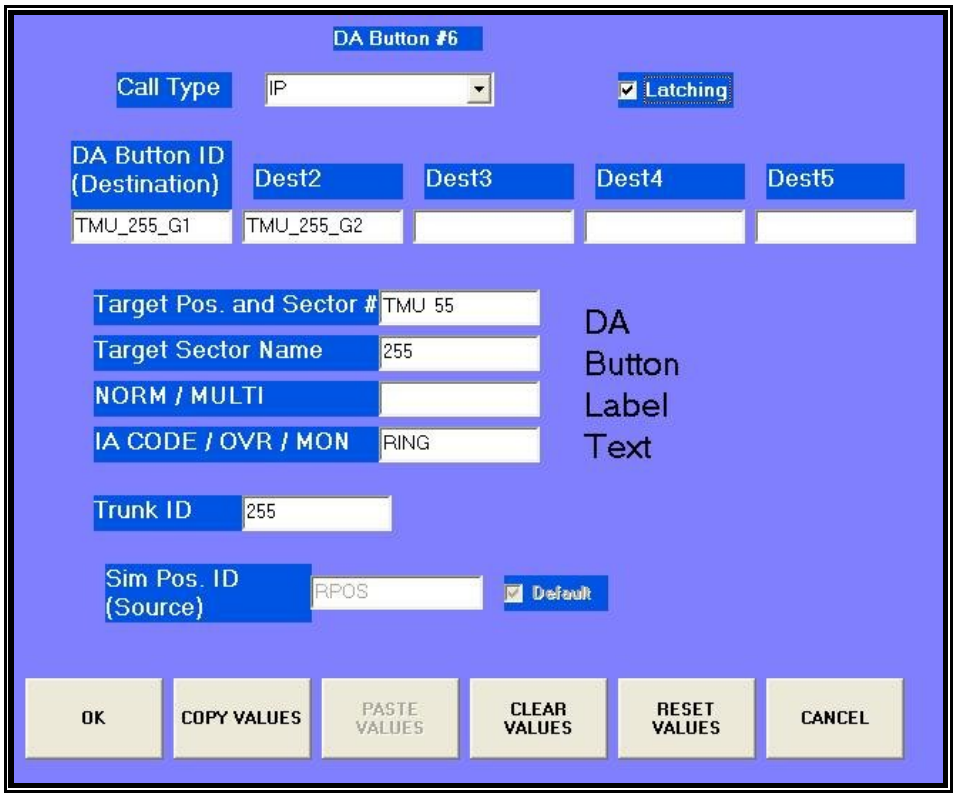


Figure 9-8 Adaptation Builder DA Button Detail Screen

When the OK control button is selected to complete data entry for the DA Button, the DA Button Detail Screen is removed. If the DA Button data was changed, the representative button on the G/G Sub-tab displays a change indicator (five red asterisks) at the top of the button (See Button #2 in figure 9-7)

As mentioned above, the copy and paste functions included in the control buttons are also available as right mouse click options (along with a cut option, which executes a copy and also deletes the detail data from that DA button) while hovering on any of the DA buttons themselves within the A/G or G/G Sub-tab. This capability for the G/G screen is depicted in Figure 9-9. The same menu is used for the A/G screens.

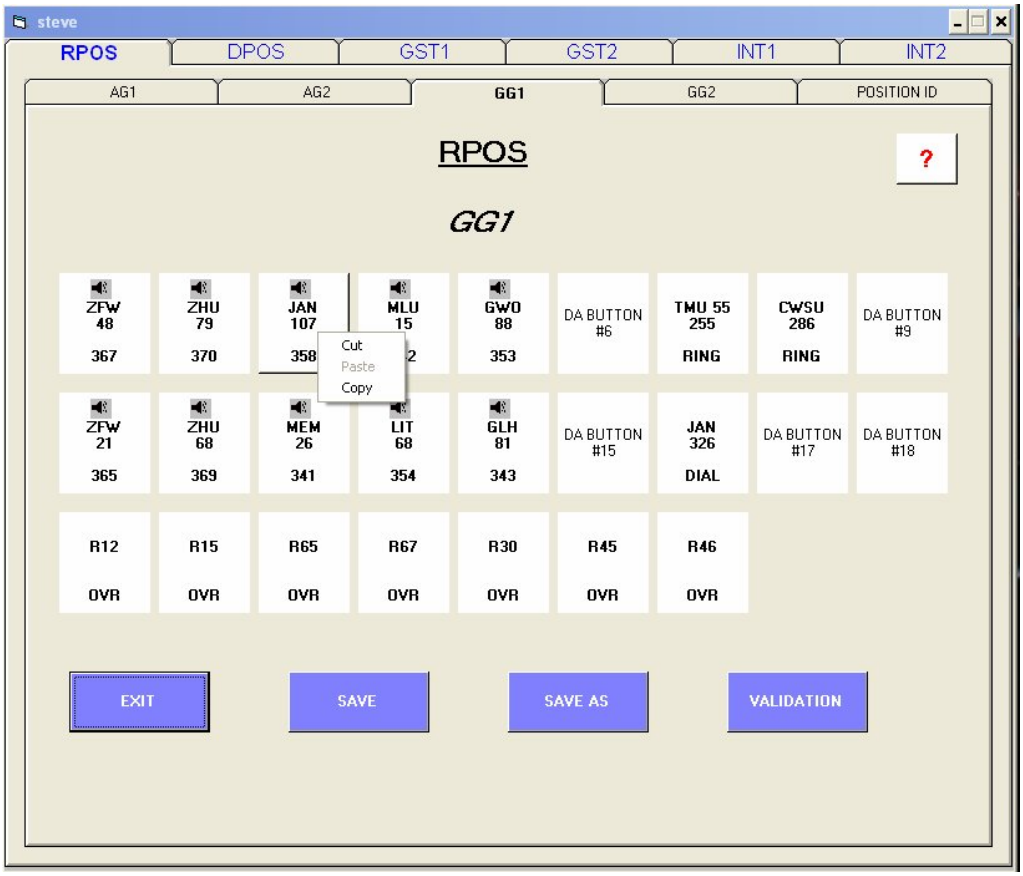


Figure 9-9 Adaptation Builder G/G1 Screen with Copy/Paste/Cut Right Mouse Click Menu

The default DA Button Detail Screen provides text boxes for entry of multiple data points that are related to the DA button. These data points are described in the following sections.

9.3.1.1.2.1 DA Button Call Type

The DA Button Detail Screen allows the user to select the DA button types. The valid types are:

- IP (Inter-Phone)
- IP Holler (Inter-Phone Holler/Shout)
- IP Dial (Inter-Phone using the VIK)
- Monitor (MON) (Voice Monitoring)

- IC (Inter-Conference)
- Override

The Monitor and Override call types also allow for the button to be latching or non-Latching (non-latching buttons are only active when they are being pressed). All other call type may only be latching.

9.3.1.1.2.2 DA Button ID (Destination)

The DA Button Detail Screen provides a textbox area for entry of DA Button ID(s) or destination(s). The DA Button ID's are limited to 12 alphanumeric characters. Based on call type there can be one or many destinations. These destinations may be entered in free form with no restriction to entry. In the case of IP DIAL calls, the user is also required to enter one or more dial codes associated with the DA Button. These dial codes must each have one or more destinations associated with them. DA Button Detail Screen provides textboxes for these entries. Figure 9-10 depicts the Adaptation Builder DA Button Detail Screen with an IP Dial call type selected. The DA Button ID is a required parameter.

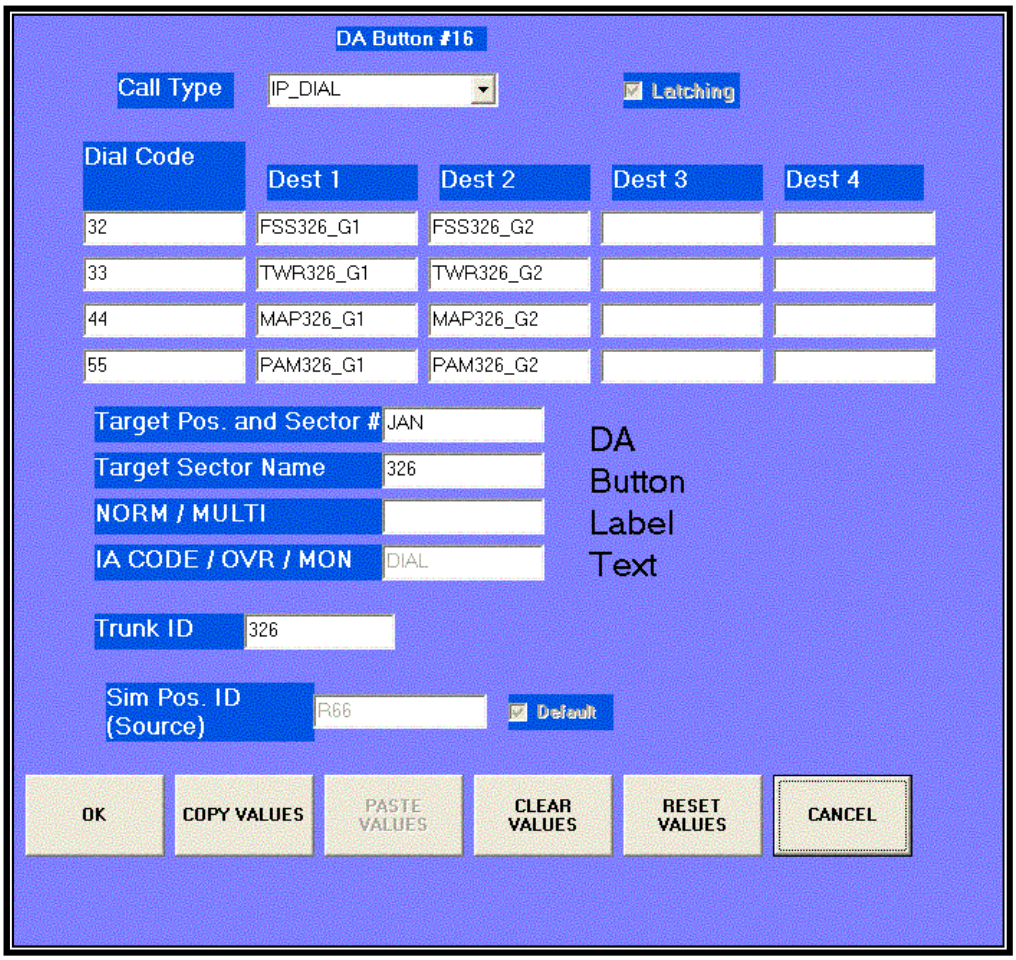


Figure 9-10 Adaptation Builder DA Button Detail Screen

9.3.1.1.2.3 DA Button Label Text

The DA Button Detail Screen provides a textbox area for entry of DA Button Label text. The screen allow for four lines of text data. Each line of label information is limited to 7 alphanumeric characters. This text will be displayed on the four line DA button label in the G/G Screen at the position indicated in the main tab. In some cases text will be pre-entered based on the call type of the DA Button (i.e. line 4 of a Monitor Call will be pre-entered as "MON"). This pre-entered data is not changeable by the user and is designed to give the buttons a look that is closer to the VSCS.

The data entered in the DA Button Label text is freeform and each line is limited to 7 characters. Note that even though the text is freeform it is recommended that the labels

are created to provide meaning to how a call will proceed if the button is used. This is especially important at the Ghost Pilot positions where the user is expected to call and answer students as various G/G entities at any given point in a scenario. An example of meaningful labeling is depicted in Figure 9-11. This Override Call button is designed Override the R Position student from the Ghost1 Pilot position. The button labeling provides detail to the Ghost position user to indicate that upon initiation of the call the R student position user will be notified that the D position (simulated by the Ghost position) at sector 12 is overriding him/her.

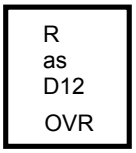


Figure 9-11 Override DA Button Labeling for Ghost Position

9.3.1.1.2.4 Trunk ID

The DA Button Detail Screen provides a textbox area for entry of a DA Call Trunk ID value. The Trunk ID is limited to 6 alphanumeric characters. This value is used for simulation of trunk line allocation for interfacility type calls. This value is only valid for entry on Inter-Phone type calls (IP, IP Holler, and IP Dial) and is a required parameter fo those call-types.

9.3.1.1.2.5 SIM Source ID

The DA Button Detail Screen provides a textbox area for entry of a DA Call Simulation Source ID. This value allows Ghost Pilot Positions to have different sources identified from button to button. This allows the Ghost Pilots to place/receive calls simulating multiple locations. The SIM Source ID is only changeable button to button for the Ghost Pilot positions. The SIM source Id defaults to the position ID value on the Ghost Pilots positions (located on the Position ID Sub-tab) if the default check box to the right of the of the SIM Source ID text box is checked. The SIM Source ID for all DA Buttons on all non-Pilot positions is the position ID assigned on the Position ID Sub-tab. The SIM Source ID is a required parameter and is limited to 6 alphanumeric characters.

9.3.1.1.3 Position ID Sub-tab

The Position ID Sub-tab, depicted in Figure 9-12, allows the user to enter position related data for each of the six positions within the adaptation set. A textbox is provided on the Position ID Sub-tab for the user to enter a name that will identify that position. This name may subsequently be used as a destination ID on DA buttons at other positions within the adaptation set.

NOTE – The Position ID value becomes the SIM Source ID for all DA Buttons on non-Pilot positions or DA Button that indicate the SIM Source ID should the default ID.

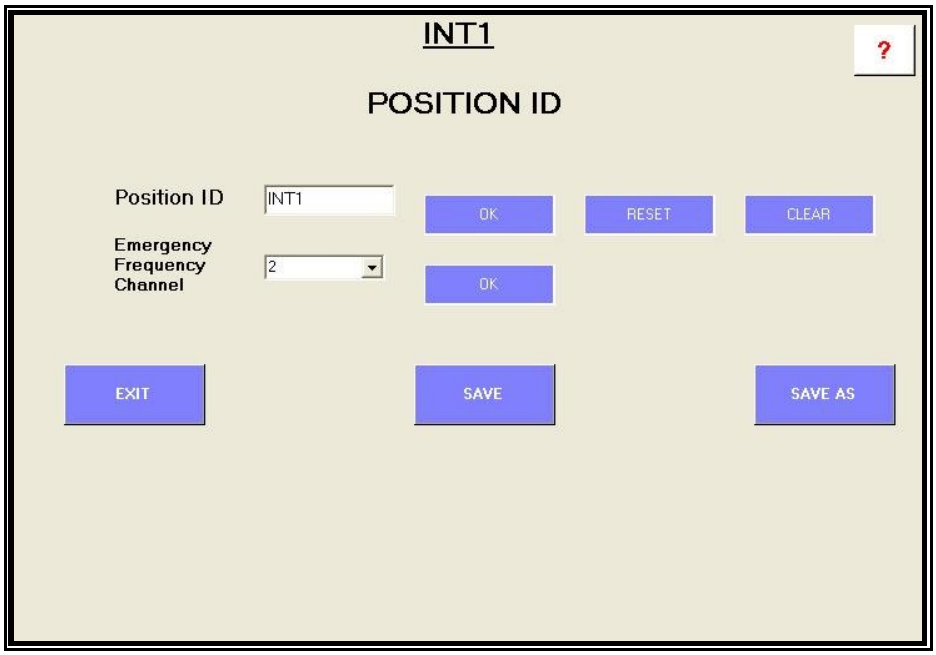


Figure 9-12 Adaptation Builder Position ID Sub-tab for Position INT1

9.3.1.1.3.1 Emergency Frequency Channel Designator

The Position ID Screen also allows the user to designate either of the two frequency Channel IDs within VCS as an emergency frequency. If any frequency channel is designated as emergency, all A/G frequency buttons, within the adaptation set, assigned to that frequency channel will be designated emergency. This will be indicated by a "checked" box for Emergency in the A/G Frequency Detail Screen (see Figure 9-6).

9.3.2 Adaptation Builder Validation Description

Because of the complexities involved in building coherent G/G communication adaptation, the VCS Adaptation Builder provides a process for validating the G/G adaptation within an adaptation set. The G/G Sub-tab provides a button (see Figure 9-8) to trigger an adaptation set validation process for G/G communication data across the entire adaptation set. Upon completion of the validation process a text report is displayed to the user using the Microsoft Notepad application. The report is depicted in Figure 9-12. The report is also saved in the adaptation set folder under the name "validation_errors.txt" (see Figure 9-13).

NOTE - The user should close the Notepad window after reviewing and fixing any errors, before running Validation again.

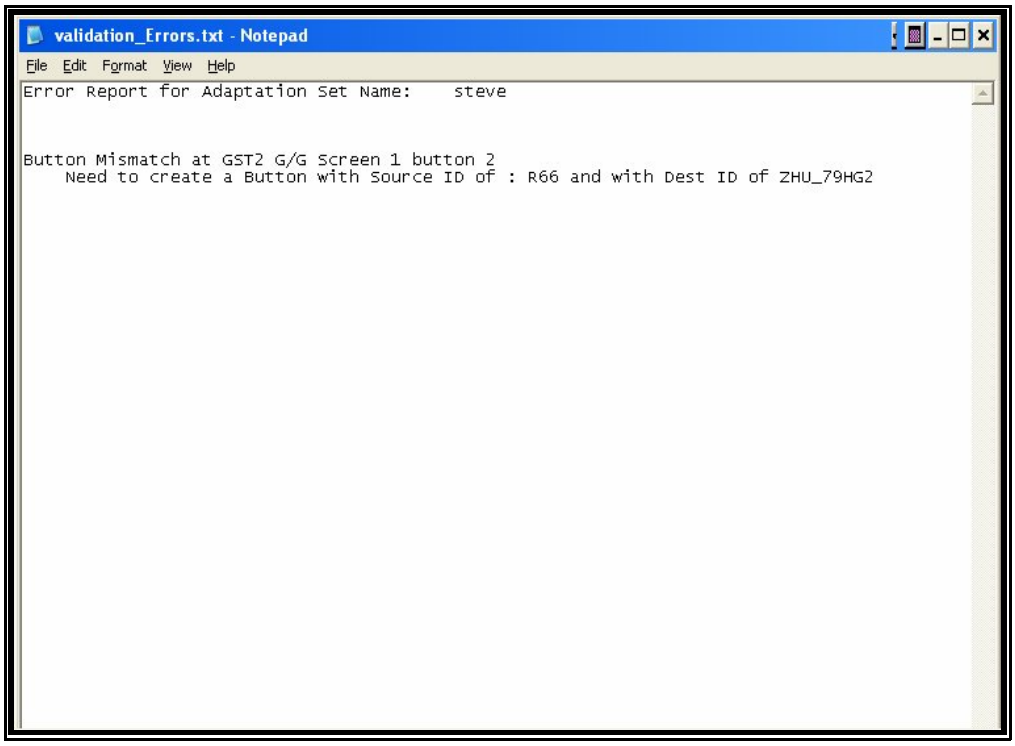


Figure 9-13 Adaptation Builder G/G Validation Report

9.3.2.1 Adaptation Builder Validation Process

The adaptation builder validation ensures that all call buttons within an adaptation set are valid. The process checks that every destination ID identified within all the DA Buttons in an adaptation set has a least one matching SIM Source ID associated with it.

Additionally, for IP, IP_DIAL, and IP_HOLLER call type buttons, the trunk IDs within two destination/Sim Source ID button matches are examined for compatibility. The trunk IDs need to be identical to have a matching IP, IP_DIAL, and IP_HOLLER call type buttons.

Figure 9-14 depicts two side by side DA Button detail screens. Arrows indicate what needs to match for two buttons to be compatible.

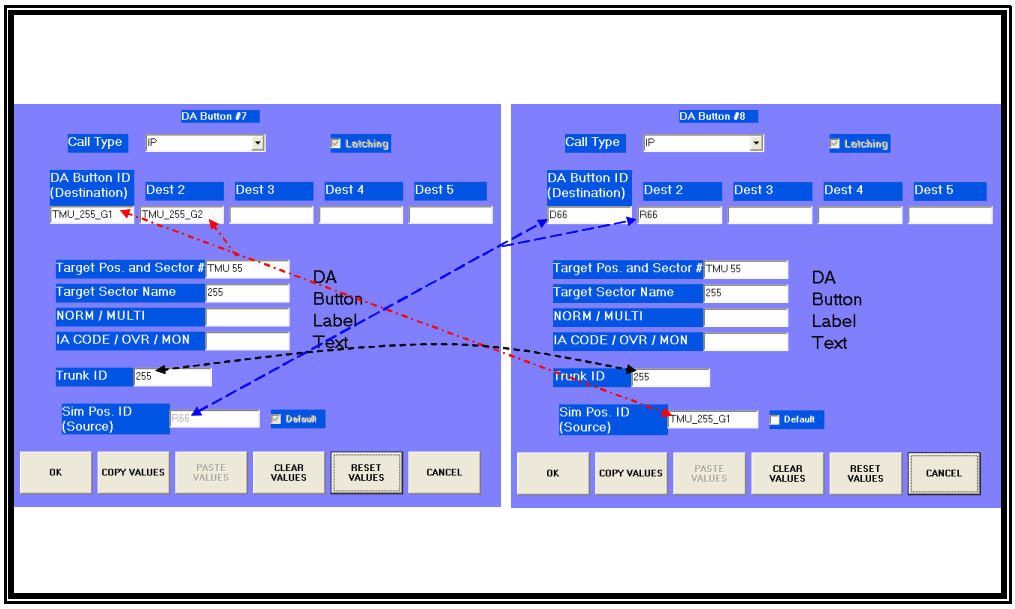


Figure 9-14 Adaptation Builder Validation Button Match Criteria

10 VCS Configuration Guide

10.1 VCS COM Server Configuration

10.1.1 Runtime Link Library Requirements

The runtime library requirements are listed in Appendix A of this document.

10.1.2 COM Server File System Configuration

The COM Server software leverages the Microsoft Windows operating system to perform many VCS Administrative operations such as adaptation loading and software executable distribution. Additionally, the operating system is used to provide remote file access to the Adaptation Builder application from the COM Server. These functions require that the file system on the COM Server to be set up very specifically. Figure 10-1 depicts the overall file system layout of the VCS COM Server.

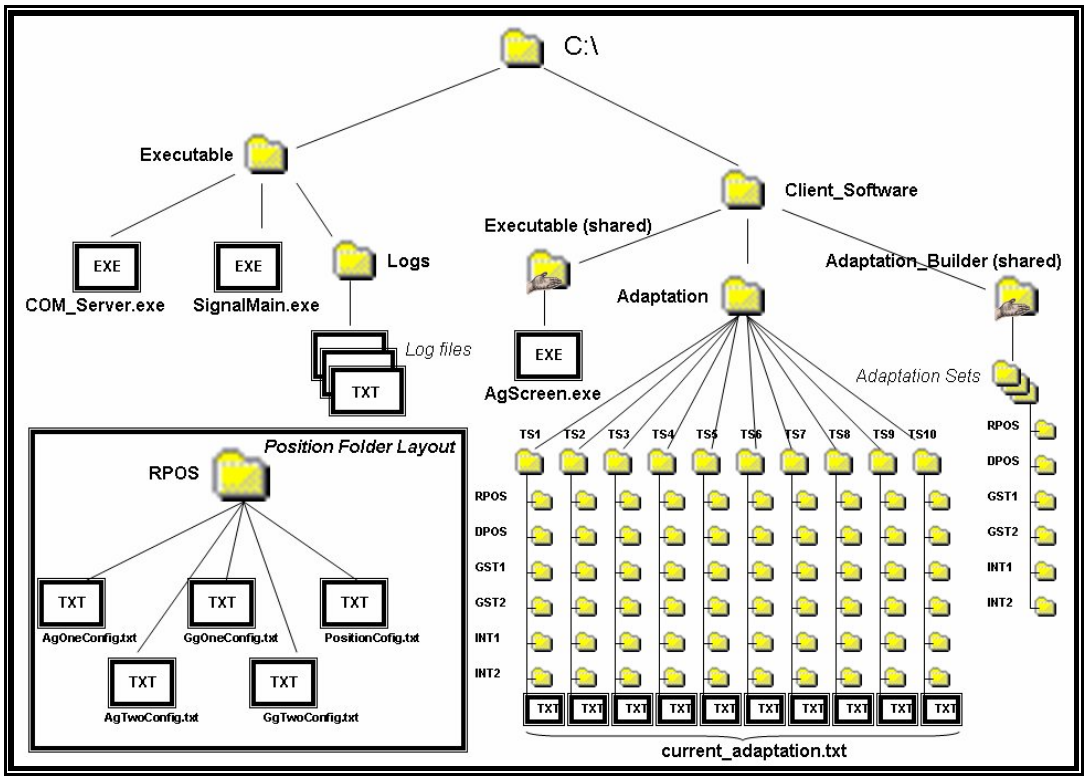


Figure 10-1 COM Server File System Layout

There are two main folders located on the C hard drive on the COM Server; the **Executable** and the **Client_Software** folders.

The Executable folder contains three executable files that together comprise the COM Server application. These are called COM_Server.exe (the main executable), SignalMain.exe (the Remote Procedure Call executable), and DistributeIATSClient.exe (the software distribution part of the COM server). A **Logs** folder is also contained under the Executable folder. The COM server writes a new text file to the logs folder every time it is initialized.

The Client_Software folder contains client related data including an executable and adaptation data for all 10 training sectors within the Lab (3 training sectors for the Mini Lab). This data breaks out into three folders under Client_Software. The folders are titled **Adaptation Builder**, **Adaptation**, and **Executable.**

The Adaptation_Builder folder contains all the loadable adaptation sets for the laboratory that the COM Server resides in. These adaptation sets are created using the Adaptation Builder Application described in Section 9.3. Each adaptation set is contained in a folder whose name is the name that is given to the adaptation set by the adaptation build process. Each adaptation set folder contains a full training sector adaptation set as described in Section 9.2. The Adaptation_Builder folder is a shared folder under the Microsoft XP Operating System. This is setup to allow the Adaptation Builder application to have permission to upload new adaptations to the folder for use in the laboratory. The VCS system does not apply substantial security to shared file systems; however the design will support a more selective user access policy to shared file system if it is deemed necessary in the future.

The Adaptation folder contains all the currently loaded adaptation sets for each VCS training sector within the laboratory. The adaptation sets break out into ten folders titled **TS1-TS10.** A full training sector adaptation set as described in Section 9.2, is contained within each training sector folder. Additionally a separate text file is contained within each training sector folder called **Current_Adaptation.txt.** This file contains one line with the name of the adaptation set that is contained within this training sector folder. This file is created/modified by the COM Server adaptation load process as described in Section 11.2.2. The name that is placed in the file is actually the name of the adaptation set folder located in the Adaptation_Builder folder that was copied into the training sector folder under the Adaptation folder during the COM Server adaptation load process.

The Client_Software Executable folder contains the currently loaded (at all clients) client executable. This executable may be distributed to the clients using the COM Server software Distribution Utility described in Section 11.2.3. The Executable folder is a shared folder under the Microsoft XP Operating System. This is setup to allow a user on the SDE to upload a new executable for distribution. The VCS system does not apply substantial security to shared file systems; however the design will support a more selective user access policy to shared file system if it is deemed necessary in the future.

10.1.2.1 PTT Chassis Setup

The PTT Chassis port allocation in each lab should follow the network switch configuration .

Each client position is allocated one chassis port. The system is organized such that the port number allocated should match the network switch port number allocated for that same client (described in Section 10.4). Additionally, the port number should be identical to the last octet number in the IP address of the client as specified in the system Hosts file (described in 11.2.3 and listed in Appendix B). For example, Client LAB1_TS1_RPOS has an IP address of 172.26.41.1 and uses port 1 on both the network switch and PTT chassis. This convention is not a functional requirement, but is used in the interest of organization.

10.2 VCS DSP Configuration

10.2.1 Main Lab DSP Hardware Setup

10.2.1.1 Break Out Box Setup Wiring

The DSP Break out Box wiring should be set up as defined in Section 7.3. Note that the break out boxes must be powered on prior to powering on the DSP.

10.2.2 VCS DSP Description Sheet Listing

The following is a listing of the VCS DSP description sheets.

- TrainingGroupA.des
- TrainingGroupB.des
- Training_GroupC.des (Lab1 and Lab2 only)
- Trainng_GroupD.des (Lab1 and Lab2 only)
- Common.des
- Viktones.des

These sheets are collected in a project files as well as a runtime environment file represented by files titled **IATS-VCS.vpj and IATS-VCS.vne.**

10.3 VCS Client Configuration

10.3.1 Runtime Link Library Requirements

The Runtime library requirements are listed in Appendix A of this document.

10.3.2 Client File System Configuration

The VCS Client file system layout is comprised of one folder called **Executable** residing in the base **C:** directory of each client. The folder contains the VCS client executable called **AgScreen.exe** and a **Logs** folder. The VCS Client writes a new text file to the logs folder every time it is initialized. The Executable folder is a shared folder under the Microsoft XP Operating System. This is setup to allow a user on the COM Server to distribute a new software executable to the client. The VCS system does not apply substantial security to shared file systems; however the design will support a more selective user access policy to shared file system if it is deemed necessary in the future.

10.4 VCS Network Switch Configuration

10.4.1 VCS Layer Two Switch Configuration

The main lab switch configuration requires that two VLANs be defined, One VLAN (VLANx2) for COM server/DSP communications and one VLAN (VLANx1) for COM server/client communications.

Four ports (ports 77-80) out of the 80 ports on the switch are configured for VLANx2. This includes ports for the COM server and three DSPs within the lab. The rest of the ports (1-76) are on VLANx1. The switch port number that a client is physically connected corresponds to that client's PTT chassis port number.

The Switch is also configured with an IP interface on the server VLAN for password protected remote access from the DSPs or the COM Server for that lab. Access is also available using the switch console port.

The MINI laboratory switch is laid out similarly to the main labs; however there are only 24 ports on the switch. VLAN x2 is comprised of ports 21-24. The remaining ports are in VLAN x1.

10.5 Adaptation Builder Configuration

10.5.1 Runtime Link Library Requirements

The Runtime library requirements are listed in Appendix A of this document.

10.5.2 Network Drive Mapping

The Adaptation Builder is set up specifically to run on the SDE machine, however it maybe run on any of the COM Server machine or a machine with network connectivity to the COM server. At startup, the Adaptation_Builder attempts to map three network drives; one drive per COM Server (per lab) across three laboratory networks. The network drives represents the shared Adaptation_Builder folder within each lab. Whichever network drives are successfully mapped, get presented to the user in the Lab Selection Dialog. If no drives are mapped successfully, the application notifies the user and then exits.

11 Operating the IATS VCS System

11.1 System Startup

11.1.1 VCS COM Server Startup

The IATS VCS COM Server Application will be started via a desktop icon at the Master Instructor Workstation. Upon selection of the icon, the COM Server application will start up and display the main form consisting of objects pertaining to all of the hardware elements of the IATS VCS. In the Figure 11-1, noteworthy is the green Server and DSP1 objects indicating that the Server is up and running and that the DSP1 is also up and running and connected to the server. There are no clients connected at startup. Client startup/shutdown is a Master Instructor capability.

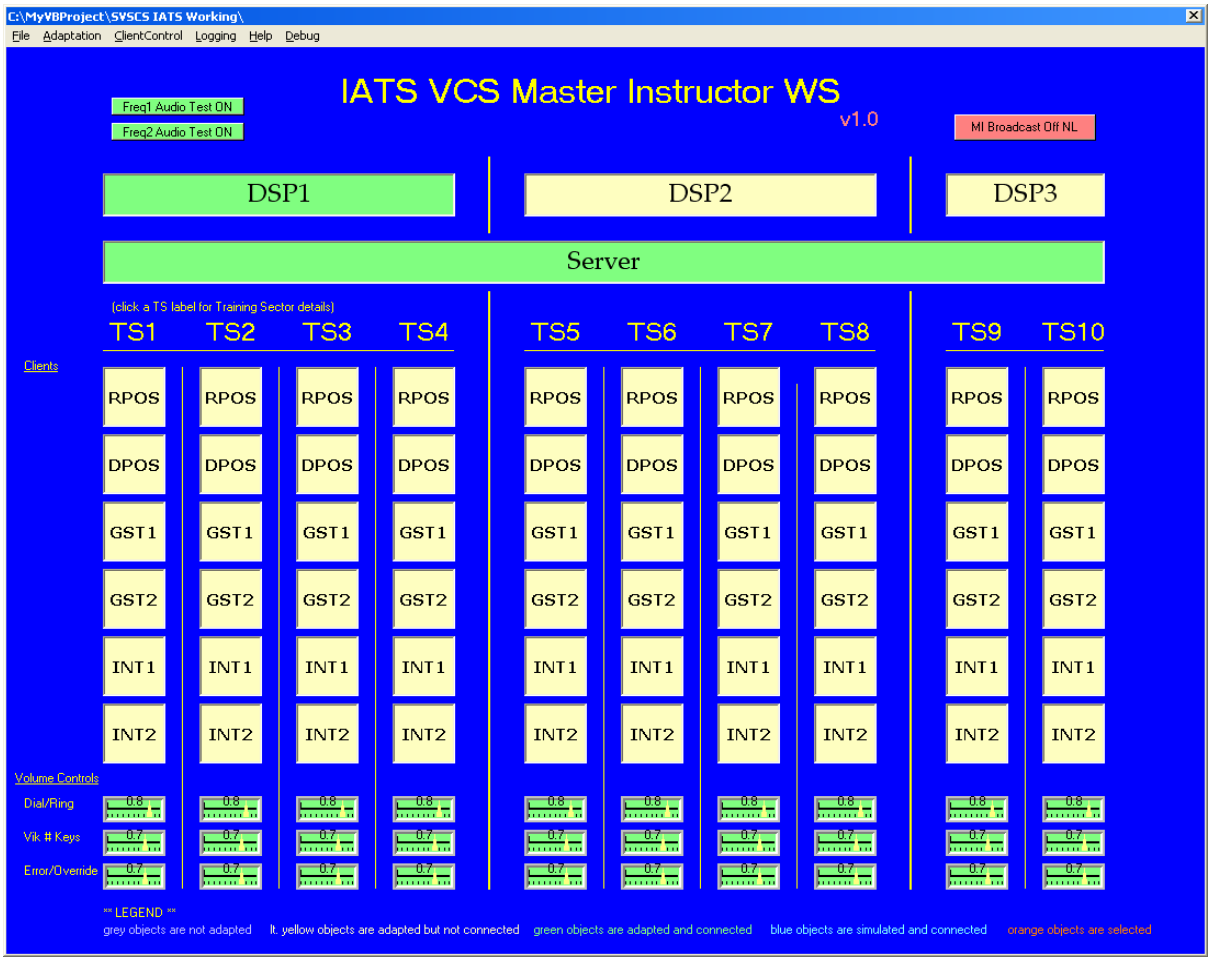
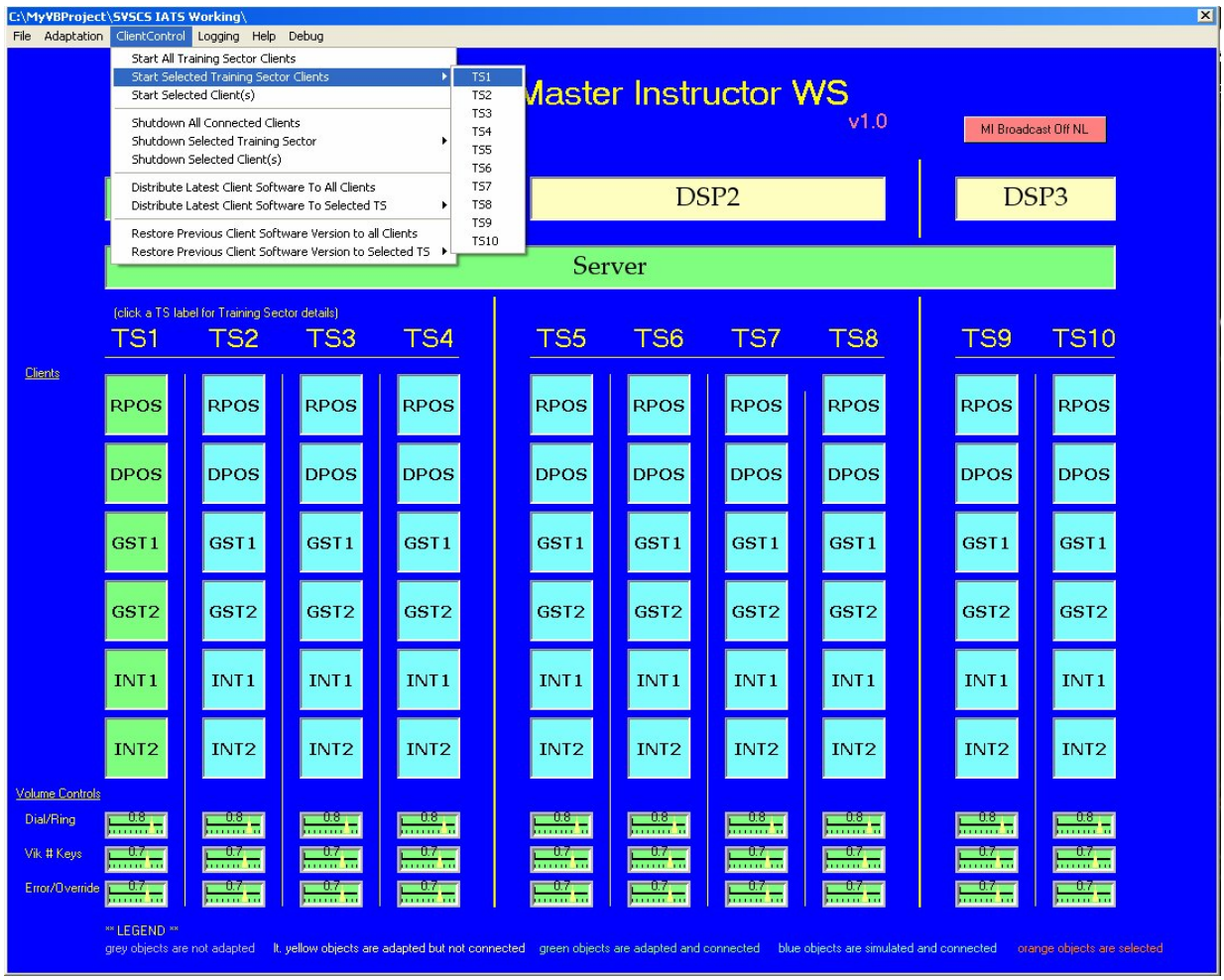


Figure 11-1 COM Server Main Dialog

11.1.2 Starting Clients

The IATS Clients are started from the Master Instructor WorkStation COM Server Application (see Figure 11-2). Main menu bar pull down menus permit start up of one client or groups of clients concurrently. Groups of clients may be selected graphically or may be specified by training sector.



11.2 System Reconfiguration

11.2.1 Start /Shutdownof VCS Clients

The IATS Clients can be started or shutdown from the Master Instructor WorkStation COM Server application (see Figure 11-3). Main menu bar pull down menus permit started or shutdown of one client or groups of clients concurrently. Groups of clients may be selected graphically or may be specified by training sector.

Figure 11-3 COM Server System Shutdown

11.2.2 Loading Client Adaptation

The IATS Clients training sectors can be loaded with a specific adaptation set prior to startup via a main menu bar pull down menu at the Master Instructor Workstation COM Server Application (see Figure 11-4).

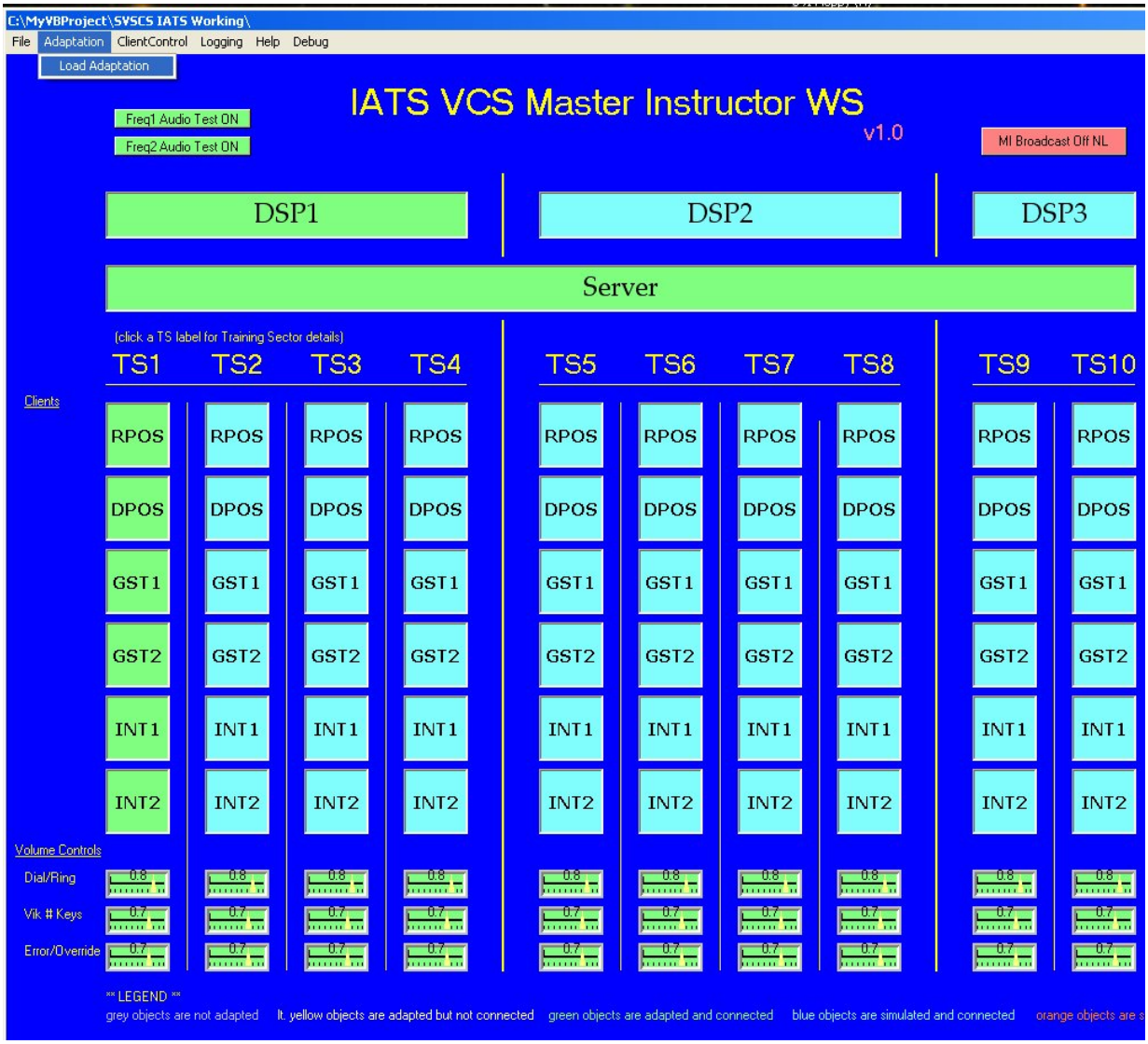


Figure 11-4 COM Server Adaptation Load

Once the ‘Training Sector Adaptation’ form, Depicted in Figure 11-5 is displayed, the user can select the Training Sector that adaptation should be loaded for by clicking on the ‘Load New Adaptation’ button for the associated training sector.

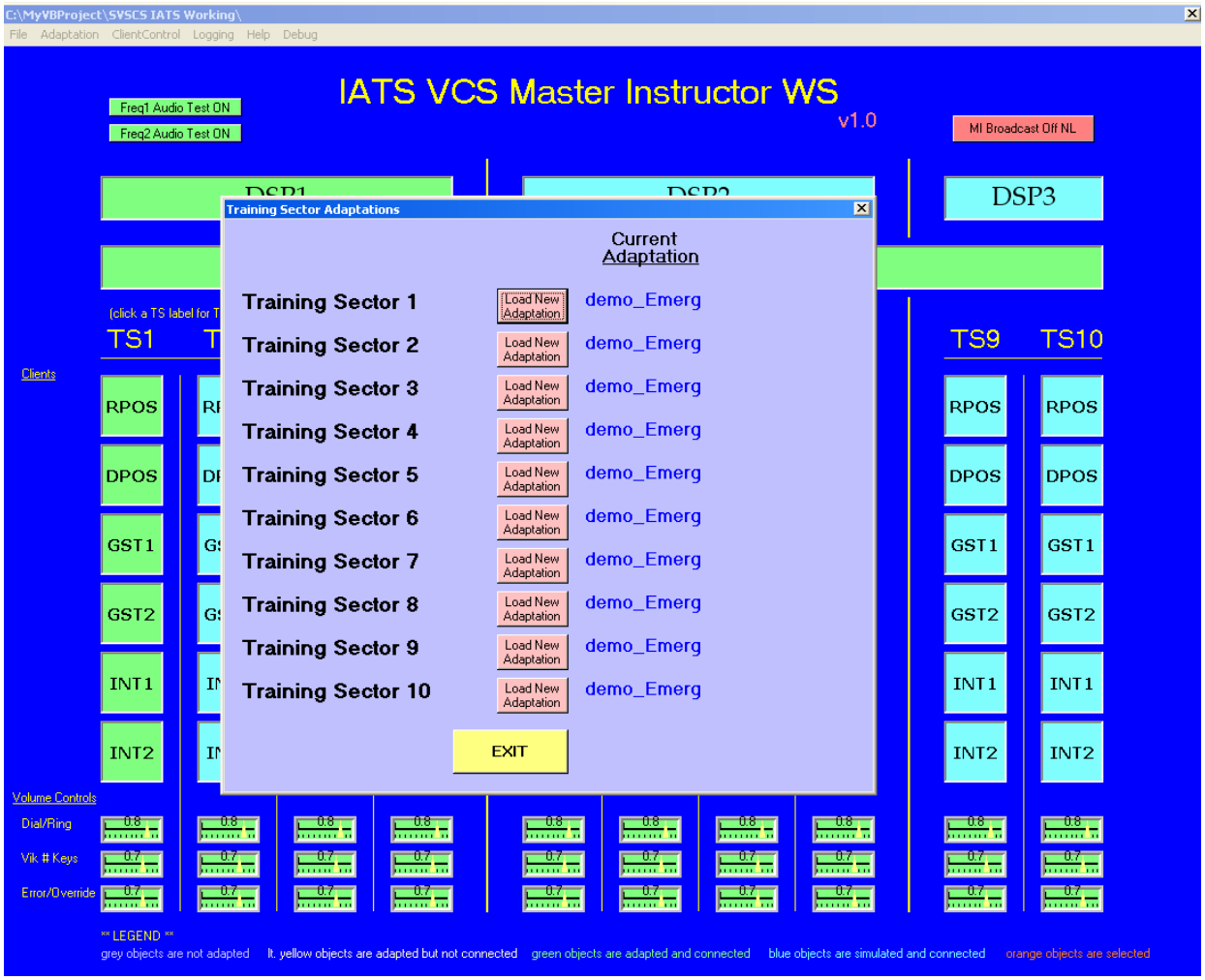


Figure 11-5 COM Server Training Sector Adaptation Dialog

Once the 'Select an Adaptation' form, depicted in Figure 11-6 is displayed, the user can select the adaptation set to preload into the training sector. When clients in the particular training sector are started, they will be configured with that specific adaptation set.

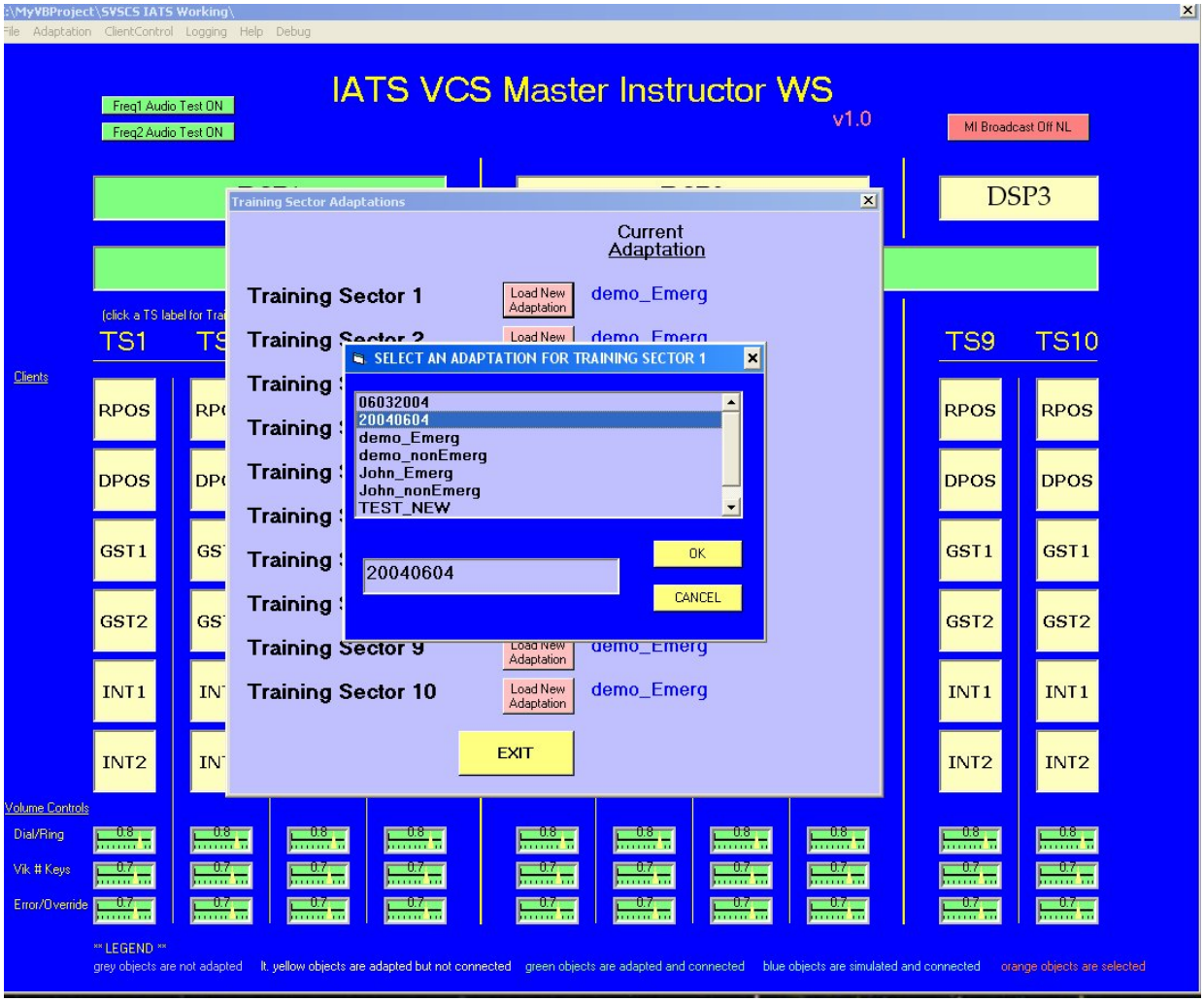


Figure 11-6 COM Server Training Sector Adaptation Selection Dialog

11.2.3 Distributing Client Software

VCS Client software can be distributed to all of the clients in one or more training sectors at a time via a main menu bar pull down menu at the Master Instructor Workstation COM Server application (see Figure 11-7). This operation also backups up the currently running executable for use in the software fallback process (Section 11.2.4).

Figure 11-7 COM Server Client Software Distribution

11.2.4 Restoring Previous Versions of Client Software

If the software that has been distributed does not run as expected, there is also a method to 'restore' (fallback to) the previous version of client software in this same pull down menu (see Figure 11-8).

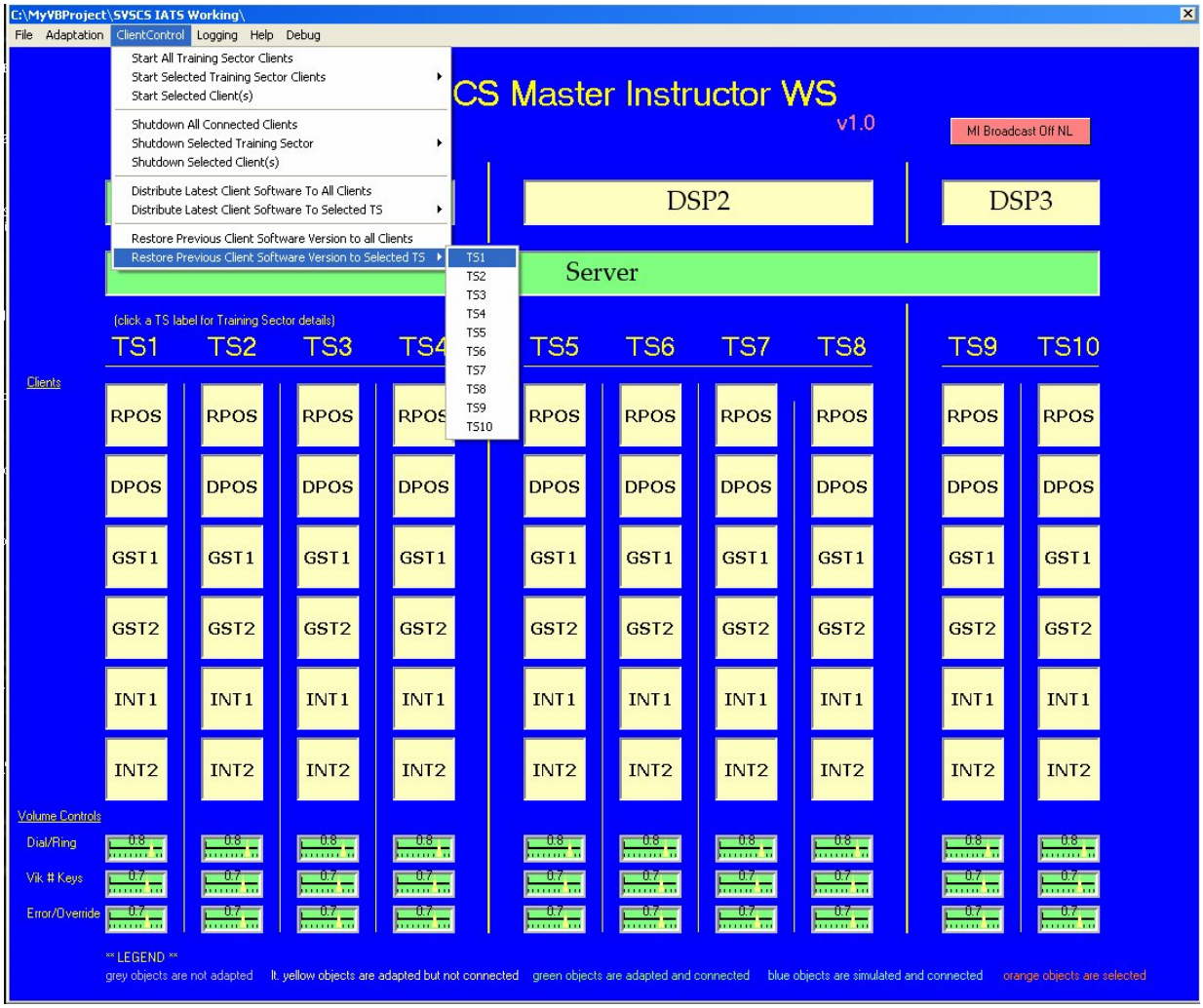


Figure 11-8 COM Server Client Software Fallback

11.3 System Logging

11.3.1 Viewing the System Log

To view the system log, pull down the 'Logging" main menu bar item and select Viewer (see Figure 11-9). A log window will appear on top of the main form showing time stamped text messages detailing the events taking place in the server.

Figure 11-9 COM Server Logging

12 Using the VCS Software Development Environment

12.1.1 Using the SDE to view and modify software

The VCS software resides on the SDE machine in a folder on the C:\ disk called **IATS_VCS_SOFTWARE.** There are five folders in the **IATS_VCS_SOFTWARE** folder called **VCS_CLIENT, COM_SERVER, ADAPTATION_BUILDER, SIGALMAIN,** and **DSP.** They represent the five primary software areas comprising the VCS system. Each of these "software area" folders may contain several dated subfolders that contain previous versions or backups of that particular software area. The VCS_CLIENT subfolder will contain a file with a ".dsw" extension that is the Visual C++ project containing the VCS Client software. The COM_SERVER subfolder will contain a file with a ".vbp" extension that is the Visual Basic project containing the VCS Com Server. The COM Server subfolder also contains an additional subfolder (called "DistributeClientSW") that contains the Software Distribution project. Within this subfolder is a file with a ".vbp" extension. This Visual Basic project contains the source code for the VCS COM Server Software Distribution Function. The ADAPTATION_BUILDER subfolder will contain a file with a ".vbp" extension that is the Visual Basic project containing the Adaptation Builder software. The SIGNALMAIN folder will contain a slightly modified version of the RPC software provided by MMAC. The DSP folder will contain the Visual Programming Language (V+) description sheets. Double-clicking on ".dsw" or ".vbp" files will initialize Visual Studio and load that particular project and the software may then be viewed, modified, and recompiled. Server and DSP software may not be executed on the SDE machine because the required interface cards are not installed. Client software and Adaptation Builder software may be executed on the SDE. DSP description sheets may only be edited on the DSP.

12.1.2 Connecting the SDE to the Mini Lab DSP

The SDE has the capability to connect to the COM Server simulating any of the adapted VCS clients within a Laboratory. This becomes essential when debugging a problem that is specific to one particular client position. The SDE client's identity must be assigned using the COM Server debug pull down menu as depicted in Figure 12-1 prior to initialization of the SDE client.

Figure 12-1 VCS COM Server SDE client Assignment

When the SDE client is initialized, the user is prompted to select which laboratory to submit a connection request to (see Figure 12-2). Therefore, the selection of the SDE's client identity must be performed at the COM Server associated with the Laboratory selected before the SDE client is initialized. When the COM Server is presented with the SDE connection request, it accommodates the request and correlates the audio interfaces using the COM Server's assigned SDE Simulated Position.

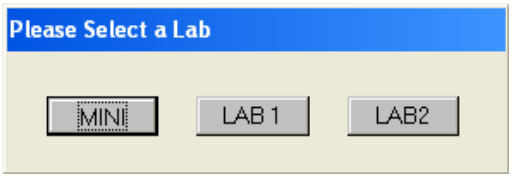


Figure 12-2 VCS SDE client Laboratory Selection Dialog

Appendix A

VCS Runtime File Listing

The following runtime files are required for execution of VCS. These files must be included in the Microsoft Windows XP System32 directory.

Visual C++/Visual Basic

Date/Time	Size	Name
06/17/1998 12:00a	929,844	MFC42D.DLL
06/17/1998 12:00a	41,013	MFCN42D.DLL
06/17/1998 12:00a	798,773	MFCO42D.DLL
08/09/1998 10:07a	94,208	MSSTKPRP.DLL
11/08/2000 03:02p	1,388,544	msvbvm60.dll
06/17/1998 12:00a	516,173	MSVCP60D.DLL
07/22/2002 12:05p	290,869	msvcrt.dll
06/17/1998 12:00a	385,100	MSVCRTD.DLL
10/23/1998 05:44p	108,560	NumLED.ocx
05/30/1998 08:00p	22,288	COMCAT.DLL
07/22/2002 12:05p	626,960	OLEAUT32.DLL
07/22/2002 12:05p	164,112	OLEPRO32.DLL
03/26/1999 01:00a	101,888	VB6STKIT.DLL
03/26/1999 01:00a	73,216	ST6UNST.EXE
07/22/2002 12:05p	16,896	STDOLE2.TLB

COTS Active X Files

Impulse Studio

Date/Time	Size	Name
03/13/2001 02:49p	140,288	Comdlg32.ocx
10/27/1999 04:05p	86,016	fpOutBar.ocx
10/23/2002 12:21a	118,784	ImpulseAboutBox.ocx
10/23/2002 01:01a	262,144	ImpulseAniLabel.ocx
07/11/2001 06:14a	61,440	ImpulseASM.dll
10/23/2002 01:02a	475,136	ImpulseButton.ocx
10/23/2002 01:03a	188,416	ImpulseCheckBox.ocx
10/23/2002 01:03a	360,448	ImpulseContainer.ocx
10/23/2002 12:21a	147,456	ImpulseFolderSelect.ocx
10/23/2002 01:04a	196,608	ImpulseFrame.ocx
10/23/2002 01:28a	1,388,544	ImpulseGlobals.dll
10/23/2002 01:05a	106,496	ImpulseHazard.ocx
10/23/2002 12:32a	147,456	ImpulseInputBox.ocx
10/23/2002 01:06a	659,456	ImpulseListBar.ocx
10/23/2002 01:07a	208,896	ImpulseMessageHook.ocx
10/23/2002 01:08a	270,336	ImpulseMixer.ocx
12/18/2000 02:18a	122,880	ImpulseMP3.dll
10/23/2002 01:12a	241,664	ImpulseMP3.ocx
10/23/2002 12:21a	585,728	ImpulseMsgBox.ocx
10/23/2002 01:21a	188,416	ImpulseOptionButton.ocx
10/23/2002 12:56a	262,144	ImpulsePassword.ocx
10/23/2002 01:23a	77,824	ImpulsePower.ocx
10/23/2002 01:23a	319,488	ImpulseScrollBars.ocx
10/23/2002 01:00a	724,992	ImpulseSplashScreen.ocx
10/23/2002 01:24a	77,824	ImpulseTimer.ocx

10/23/2002	01:25a	188,416	ImpulseTrayIcon.ocx
10/23/2002	01:25a	73,728	ImpulseWindowFlash.ocx
10/23/2002	01:26a	344,064	ImpulseWizard.ocx
05/01/2000	10:22p	1,536	ISWin32.tlb

Farpoint

Date/Time		Size	Name
10/11/2001	12:28p	224,048	Btn16d20.dll
10/11/2001	12:37p	34,816	Btn16d20.lib
10/05/2001	11:04a	426	btn32a20.dep
12/17/2003	03:02p	116,224	btn32a20.oca
10/11/2001	12:28p	419,520	btn32a20.ocx
10/11/2001	12:28p	225,280	Btn32d20.dll
10/11/2001	12:37p	38,912	Btn32D20.lib
10/11/2001	12:28p	322,064	Btnvbx20.vbx
07/01/2003	01:47p	765,952	BtWizard.dll
07/01/2003	01:29p	102,400	BTXPPanel.dll
12/17/2003	03:02p	31,232	btxppanel.oca
07/01/2003	01:29p	3,780	BTXPPanel.tlb
07/01/2003	01:29p	24,576	BtXpShell.dll

GMS

Date/Time		Size	Name
11/30/1998	07:26a	214,520	Knob.ocx
10/23/1998	05:27p	119,288	LED.ocx
01/21/1999	11:31a	205,832	Lgauge.ocx
10/23/1998	05:28p	99,840	Odometer.ocx
02/01/1999	07:19p	302,088	Strip.ocx

11/20/1998	02:44p	110,096	Toggle.ocx
10/23/1998	05:28p	138,240	Percent.ocx
07/22/2002	12:05p	143,632	ASYCFILT.DLL
07/22/2002	12:05p	74,810	atl.dll
06/26/2001	04:39p	151,601	scrrun.dll
10/23/1998	05:45p	181,776	Selector.ocx
01/07/1999	09:19a	187,392	Slider.ocx
01/21/1999	11:32a	222,224	AGauge.ocx

Appendix B

VCS Hosts File Listing

```
##################### LAB 1 ###################################
#LAB1 - VLAN 1
172.26.41.100 LAB1-SERVER-MAIN

#LAB1 - VLAN 2
172.26.42.100 LAB1-SERVER-COMM
172.26.42.101 LAB1-DSP1
#172.26.42.102 LAB1-DSP2
#172.26.42.103 LAB1-DSP3

#Simulated DSPs
#172.26.41.175 LAB1-DSP1
172.26.41.175 LAB1-DSP2
172.26.41.175 LAB1-DSP3

#LAB1 - CLIENTS
172.26.41.1 LAB1-TS1-RPOS
172.26.41.2 LAB1-TS1-DPOS
172.26.41.3 LAB1-TS1-GST1
172.26.41.4 LAB1-TS1-GST2
172.26.41.5 LAB1-TS1-INT1
172.26.41.6 LAB1-TS1-INT2
```

172.26.41.7 LAB1-TS2-RPOS

172.26.41.8 LAB1-TS2-DPOS

172.26.41.9 LAB1-TS2-GST1

172.26.41.10 LAB1-TS2-GST2

172.26.41.11 LAB1-TS2-INT1

172.26.41.12 LAB1-TS2-INT2

172.26.41.13 LAB1-TS3-RPOS

172.26.41.14 LAB1-TS3-DPOS

172.26.41.15 LAB1-TS3-GST1

172.26.41.16 LAB1-TS3-GST2

172.26.41.17 LAB1-TS3-INT1

172.26.41.18 LAB1-TS3-INT2

172.26.41.19 LAB1-TS4-RPOS

172.26.41.20 LAB1-TS4-DPOS

172.26.41.21 LAB1-TS4-GST1

172.26.41.22 LAB1-TS4-GST2

172.26.41.23 LAB1-TS4-INT1

172.26.41.24 LAB1-TS4-INT2

172.26.41.25 LAB1-TS5-RPOS

172.26.41.26 LAB1-TS5-DPOS

172.26.41.27 LAB1-TS5-GST1

172.26.41.28 LAB1-TS5-GST2

172.26.41.29 LAB1-TS5-INT1

172.26.41.30 LAB1-TS5-INT2

172.26.41.31 LAB1-TS6-RPOS

172.26.41.32 LAB1-TS6-DPOS

172.26.41.33 LAB1-TS6-GST1

172.26.41.34 LAB1-TS6-GST2

172.26.41.35 LAB1-TS6-INT1

172.26.41.36 LAB1-TS6-INT2

172.26.41.37 LAB1-TS7-RPOS

172.26.41.38 LAB1-TS7-DPOS

172.26.41.39 LAB1-TS7-GST1

172.26.41.40 LAB1-TS7-GST2

172.26.41.41 LAB1-TS7-INT1

172.26.41.42 LAB1-TS7-INT2

172.26.41.43 LAB1-TS8-RPOS

172.26.41.44 LAB1-TS8-DPOS

172.26.41.45 LAB1-TS8-GST1

172.26.41.46 LAB1-TS8-GST2

172.26.41.47 LAB1-TS8-INT1

172.26.41.48 LAB1-TS8-INT2

172.26.41.49 LAB1-TS9-RPOS

172.26.41.50 LAB1-TS9-DPOS

172.26.41.51 LAB1-TS9-GST1

172.26.41.52 LAB1-TS9-GST2

172.26.41.53 LAB1-TS9-INT1

172.26.41.54 LAB1-TS9-INT2

172.26.41.55 LAB1-TS10-RPOS

172.26.41.56 LAB1-TS10-DPOS

172.26.41.57 LAB1-TS10-GST1

172.26.41.58 LAB1-TS10-GST2

172.26.41.59 LAB1-TS10-INT1

172.26.41.60 LAB1-TS10-INT2

172.26.41.61 LAB1-SDE #SDE NIC 1

##################### LAB 2 ###################################

#LAB2 - VLAN 1

172.26.51.100 LAB2-SERVER-CLIENT

#LAB2 - VLAN 2

172.26.52.100 LAB2-SERVER-DSP

172.26.52.101 LAB2-DSP1

172.26.52.102 LAB2-DSP2

172.26.52.103 LAB2-DSP3

#LAB2 - CLIENTS

172.26.51.1 LAB2-TS1-RPOS

172.26.51.2 LAB2-TS1-DPOS

172.26.51.3 LAB2-TS1-GST1

172.26.51.4 LAB2-TS1-GST2

172.26.51.5 LAB2-TS1-INT1

172.26.51.6 LAB2-TS1-INT2

172.26.51.7 LAB2-TS2-RPOS

172.26.51.8 LAB2-TS2-DPOS

172.26.51.9 LAB2-TS2-GST1

172.26.51.10 LAB2-TS2-GST2

172.26.51.11 LAB2-TS2-INT1

172.26.51.12 LAB2-TS2-INT2

172.26.51.13 LAB2-TS3-RPOS

172.26.51.14 LAB2-TS3-DPOS

172.26.51.15 LAB2-TS3-GST1

172.26.51.16 LAB2-TS3-GST2

172.26.51.17 LAB2-TS3-INT1

172.26.51.18 LAB2-TS3-INT2

172.26.51.19 LAB2-TS4-RPOS

172.26.51.20 LAB2-TS4-DPOS

172.26.51.21 LAB2-TS4-GST1

172.26.51.22 LAB2-TS4-GST2

172.26.51.23 LAB2-TS4-INT1

172.26.51.24 LAB2-TS4-INT2

172.26.51.25 LAB2-TS5-RPOS

172.26.51.26 LAB2-TS5-DPOS

172.26.51.27 LAB2-TS5-GST1

172.26.51.28 LAB2-TS5-GST2

172.26.51.29 LAB2-TS5-INT1

172.26.51.30 LAB2-TS5-INT2

172.26.51.31 LAB2-TS6-RPOS

172.26.51.32 LAB2-TS6-DPOS

172.26.51.33 LAB2-TS6-GST1

172.26.51.34 LAB2-TS6-GST2

172.26.51.35 LAB2-TS6-INT1

172.26.51.36 LAB2-TS6-INT2

172.26.51.37 LAB2-TS7-RPOS
172.26.51.38 LAB2-TS7-DPOS
172.26.51.39 LAB2-TS7-GST1
172.26.51.40 LAB2-TS7-GST2
172.26.51.41 LAB2-TS7-INT1
172.26.51.42 LAB2-TS7-INT2

172.26.51.43 LAB2-TS8-RPOS
172.26.51.44 LAB2-TS8-DPOS
172.26.51.45 LAB2-TS8-GST1
172.26.51.46 LAB2-TS8-GST2
172.26.51.47 LAB2-TS8-INT1
172.26.51.48 LAB2-TS8-INT2

172.26.51.49 LAB2-TS9-RPOS
172.26.51.50 LAB2-TS9-DPOS
172.26.51.51 LAB2-TS9-GST1
172.26.51.52 LAB2-TS9-GST2
172.26.51.53 LAB2-TS9-INT1
172.26.51.54 LAB2-TS9-INT2

172.26.51.55 LAB2-TS10-RPOS
172.26.51.56 LAB2-TS10-DPOS
172.26.51.57 LAB2-TS10-GST1
172.26.51.58 LAB2-TS10-GST2
172.26.51.59 LAB2-TS10-INT1
172.26.51.60 LAB2-TS10-INT2

172.26.51.61 LAB2-SDE #SDE NIC 2

##################### MINI LAB ##############################

#MINI LAB - VLAN1

172.26.61.100 MINI-SERVER-CLIENT

#MINI LAB - VLAN2

172.26.62.100 MINI-SERVER-DSP

172.26.62.101 MINI-DSP1

#MINI LAB - CLIENTS

172.26.61.1 MINI-TS1-RPOS

172.26.61.2 MINI-TS1-DPOS

172.26.61.3 MINI-TS1-GST1

172.26.61.4 MINI-TS1-GST2

172.26.61.5 MINI-TS1-INT1

172.26.61.6 MINI-TS1-INT2

172.26.61.7 MINI-TS2-RPOS

172.26.61.8 MINI-TS2-DPOS

172.26.61.9 MINI-TS2-GST1

172.26.61.10 MINI-TS2-GST2

172.26.61.11 MINI-TS2-INT1

172.26.61.12 MINI-TS2-INT2

172.26.61.13 MINI-TS3-RPOS

172.26.61.14 MINI-TS3-DPOS

172.26.61.15 MINI-TS3-GST1

172.26.61.16 MINI-TS3-GST2

172.26.61.17 MINI-TS3-INT1

172.26.61.18 MINI-TS3-INT2

172.26.61.19 MINI-TS4-RPOS

172.26.61.20 MINI-TS4-DPOS

172.26.61.21 MINI-TS4-GST1

172.26.61.22 MINI-TS4-GST2

172.26.61.23 MINI-TS4-INT1

172.26.61.24 MINI-TS4-INT2

172.26.61.25 MINI-TS5-RPOS

172.26.61.26 MINI-TS5-DPOS

172.26.61.27 MINI-TS5-GST1

172.26.61.28 MINI-TS5-GST2

172.26.61.29 MINI-TS5-INT1

172.26.61.30 MINI-TS5-INT2

172.26.61.31 MINI-TS6-RPOS

172.26.61.32 MINI-TS6-DPOS

172.26.61.33 MINI-TS6-GST1

172.26.61.34 MINI-TS6-GST2

172.26.61.35 MINI-TS6-INT1

172.26.61.36 MINI-TS6-INT2

172.26.61.37 MINI-TS7-RPOS

172.26.61.38 MINI-TS7-DPOS

172.26.61.39 MINI-TS7-GST1

172.26.61.40 MINI-TS7-GST2

172.26.61.41 MINI-TS7-INT1

172.26.61.42 MINI-TS7-INT2

172.26.61.43 MINI-TS8-RPOS

172.26.61.44 MINI-TS8-DPOS

172.26.61.45 MINI-TS8-GST1

172.26.61.46 MINI-TS8-GST2

172.26.61.47 MINI-TS8-INT1

172.26.61.48 MINI-TS8-INT2

172.26.61.49 MINI-TS9-RPOS

172.26.61.50 MINI-TS9-DPOS

172.26.61.51 MINI-TS9-GST1

172.26.61.52 MINI-TS9-GST2

172.26.61.53 MINI-TS9-INT1

172.26.61.54 MINI-TS9-INT2

172.26.61.55 MINI-TS10-RPOS

172.26.61.56 MINI-TS10-DPOS

172.26.61.57 MINI-TS10-GST1

172.26.61.58 MINI-TS10-GST2

172.26.61.59 MINI-TS10-INT1

172.26.61.60 MINI-TS10-INT2

172.26.61.61 MINI-SDE #SDE NIC 3

Appendix C

VCS Client Adaptation Files Example

The following is a listing of A/G, G/G, and Position ID adaptation text files.

AgOneConfig.txt:

PosId:1

FrqId:132.500

FrqName:GWO

FrqChannel:1

FrqType:NonEmergency

RemoteMute:no

BUEC:no

PTTPreempt:no

GroupMaintenance:no

MainStbyXMTR:no

MainStbyRcvr:no

XCouple:no

GroupControl:no

PosId:4

FrqId:259.100

FrqName:GWO

FrqChannel:1

FrqType:NonEmergency

RemoteMute:no

BUEC:no

PTTPreempt:no

GroupMaintenance:no

MainStbyXMTR:no

MainStbyRcvr:no

XCouple:no

GroupControl:no

PosId:7

FrqId:132.500

FrqName:JAN

FrqChannel:1

FrqType:NonEmergency

RemoteMute:no

BUEC:no

PTTPreempt:no

GroupMaintenance:no

MainStbyXMTR:no

MainStbyRcvr:no

XCouple:no

GroupControl:no

PosId:9

FrqId:243.000

FrqName:GWO

FrqChannel:2

FrqType:Emergency

RemoteMute:no

BUEC:no

PTTPreempt:no

GroupMaintenance:no

MainStbyXMTR:no

MainStbyRcvr:no

XCouple:no

GroupControl:no

PosId:10

FrqId:259.100

FrqName:JAN

FrqChannel:1

FrqType:NonEmergency

RemoteMute:no

BUEC:no

PTTPreempt:no

GroupMaintenance:no

MainStbyXMTR:no

MainStbyRcvr:no

XCouple:no

GroupControl:no

PosId:12

FrqId:121.500

FrqName:GWO

FrqChannel:2

FrqType:Emergency

RemoteMute:no

BUEC:no

PTTPreempt:no

GroupMaintenance:no

MainStbyXMTR:no

MainStbyRcvr:no

XCouple:no

GroupControl:no

GgOneConfig.txt:

1

ZFW_48HG1:ZFW_48HG2

ZFW:48::367

Latching

IP

Holler

NonDial

48

R66

2

ZHU_79HG1:ZHU_79HG2

ZHU:79::370

Latching

IP

Holler

NonDial

79

R66

3

JAN_107HG1:JAN_107HG2

JAN:107::358

Latching

IP

Holler

NonDial

107

R66

4

MLU_15HG1:MLU_15HG2

MLU:15::342

Latching

IP

Holler

NonDial

15

R66

5

GWO_88HG1:GWO_88HG2

GWO:88::353

Latching

IP

Holler

NonDial

88

R66

7

TMU_255_G1:TMU_255_G2

TMU 55:255::RING

Latching

IP

NonHoller

NonDial

255

R66

8

WSU_268_G1:WSU_268_G2

CWSU:286::RING

Latching

IP

NonHoller

NonDial

268

R66

10

ZFW_21HG1:ZFW_21HG2

ZFW:21::365

Latching

IP

Holler

NonDial

21

R66

11

ZHU_68HG1:ZHU_68HG2

ZHU:68::369

Latching

IP

Holler

NonDial

68

R66

12

MEM_26HG1:MEM_26HG2

MEM:26::341

Latching

IP

Holler

NonDial

26

R66

13

LIT_68HG1:LIT_68HG2

LIT:68::354

Latching

IP

Holler

NonDial

68

R66

14

GLH_81HG1:GLH_81HG2

GLH:81::343

Latching

IP

Holler

NonDial

81

R66

16

FSS326_G1-32:FSS326_G2-32:TWR326_G1-33:TWR326_G2-33

JAN:326::DIAL

Latching

IP

NonHoller

Dial

326

R66

19

R12

R12:::OVR

Latching

OVERRIDE

NonHoller

NonDial

NoTrunk

R66

20

R15

R15:::OVR

Latching

OVERRIDE

NonHoller

NonDial

NoTrunk

R66

21

R65

R65:::OVR

Latching

OVERRIDE

NonHoller

NonDial

NoTrunk

R66

22

R67

R67:::OVR

Latching

OVERRIDE

NonHoller

NonDial

NoTrunk

R66

23

R30

R30:::OVR

Latching

OVERRIDE

NonHoller

NonDial

NoTrunk

R66

24

R45

R45:::OVR

Latching

OVERRIDE

NonHoller

NonDial

NoTrunk

R66

25

R46

R46:::OVR

Latching

OVERRIDE

NonHoller

NonDial

NoTrunk

R66

PositionConfig.txt:

PosID:R66

PosType:ATC

Appendix D

VCS Troubleshooting Guide

TBD

Appendix E

VCS Ground to Ground Call Type Definitions

The following describes the VCS G/G call characteristics for all call types available in VCS.

IC Direct Calls

- Upon initiation
    - Source position DA Button blinks (green) and rings in the headset
    - Destination position DA Button blinks (orange) and rings in the loudspeaker
    - Third-party positions receive call placed indicator (orange bar) on all DA Buttons with the source or destination position indicated as the destination within the button (buttons are locked)
- Upon activation
    - Source position DA Button is solid green, ringing has stopped and a point-to-point audio connection is established to the destination
    - Destination position DA Button is solid green, ringing has stopped and a point-to-point audio connection is established to the source
    - Third-party positions receive call in use indicators (green bar) on all DA Buttons with the source or destination position indicated as the destination within the button (buttons are locked)
- Removal
    - An IC call may be removed by selecting the particular DA IC Button or by pressing the RLS Button at the source or destination

IC Override Calls

- Upon initiation/activation
    - Source position DA OVR Button is solid green
    - Destination position OVR indicator area displays Source Position ID
    - Destination position microphone side-tone is “hot” (overridden PTT)
    - Destination position microphone is routed to the Source headset and is “hot”
    - Destination position audio output (i.e. what is output to the Destination headset) is routed to the the Source headset
    - Source microphone is routed to the Destination headset
- Removal
    - An Override call may be removed at the source by selecting the particular DA OVR Button or by pressing the RLS Button

IC Monitor Calls

- Upon initiation/activation
    - Source position DA MON Button is solid green
    - Destination position microphone is routed to the Source headset. If the Source is an Instructor position then the microphone output is "hot"
    - If the Source is an Instructor position then the Destination position microphone side-tone is "hot" (overridden PTT)
    - Destination position output (i.e. what is output to the Destination headset) is routed to the the Source headset

Note – Multiple monitor calls may be in effect at a Source position at the same time

- Removal
    - A Monitor call may be removed by selecting the particular DA MON Button or by pressing the RLS Button
    - Monitor is also suspended when another G/G call type is initiated only to be resumed when that call is released

IP Direct Calls

- Upon initiation
    - Source position DA Button solid green (indicating allocation of a trunk line) and rings in the headset
    - Destination positions DA Buttons blink (orange) and ring in the loudspeaker
    - Third-party positions receive call placed indicator (orange bar) on all DA Buttons with the same trunk ID (buttons are locked)
- Upon activation
    - Source position DA Button is solid green, ringing has stopped and a point-to-point audio connection is established to the answering Destination
    - Answering Destination position DA Button is solid green, ringing has stopped at all destination loudspeakers and a point-to-point audio connection is established to the source at the answering Destination
    - Third-party positions (including unanswered Destinations) receive call-in-use indicators (green bar) on DA Buttons with the with the same trunk ID. These buttons allow positions to join or leave an active call
- Removal
    - Call is removed only after ALL positions have left the trunk (i.e. released from the call).

IP Holler Calls

- Upon initiation

    - Source position DA Button solid green (indicating allocation of a trunk line)
    - Destination positions DA Buttons blink (orange) and Source microphone is routed to the Destination loudspeakers
    - Third-party positions receive call placed indicator (orange bar) on all DA Buttons with the same trunk ID (buttons are locked)
- Upon activation
    - Source position DA Button is solid green and a point-to-point audio connection is established to the answering Destination
    - Answering Destination position DA Button is solid green, Source microphone is removed from the all Destination loud speakers and a point-to-point audio connection is established to the source at the answering Destination
    - Third-party positions (including unanswered Destinations) receive call in use indicators (green bar) on DA Buttons with the with the same trunk ID. These buttons allow positions to join or leave an active call
- Removal
    - Call is removed only after ALL positions have left the trunk (i.e. released from the call)

IP Dial Calls

- Upon initiation
    - Source position DA Button solid green (indicating allocation of a trunk line), ring in the headset, and the software VIK Screen is displayed
    - The IP call is initiated upon entry of a valid dial code at the VIK
    - Destination position DA Button blinks (orange) and ring in the loudspeaker
    - Third-party positions receive call placed indicator (orange bar) on all DA Buttons with the same trunk ID (buttons are locked)
- Upon activation
    - Source position DA Button is solid green, ringing has stopped, and a point-to-point audio connection is established to the destination
    - Destination position DA Button is solid green, ringing has stopped, and a point-to-point audio connection is established to the source
    - Third-party positions receive call in use indicators (green bar) on DA Buttons with the with the same trunk ID. These buttons allow positions to join or leave an active call
- Removal
    - Call is removed only after ALL positions have left the trunk (i.e. released from the call)

Appendix F

VCS COM Server performance Report

Performance testing was conducted on the COM Server and client positions at the I2F over a number of weekends. A network traffic and bandwith sniffer was used to test the load imposed by the COM server on the TCP/IP networks it uses to communicate with the clients and the DSP. Additionally, Microsoft's Management Console that comes with Windows XP was used to monitor performance objects inside of the COM Server and IATS Clients. The Miscrosoft Management Console (MMC) is accessed by going to Start=>Control Panel=>Administrative Tools=>Performance. A window appears that displays the real-time monitoring of default performance objects and counters. These default objects include Memory, Physical Disk, and Processor. The default counters in each include Pages/sec, Avg. Disk Queue Length, and % Processor Time, respectively. In addition to these defaults, the I2F test group used the Available MBytes counter from the Memory performance object, the Datagrams Received Header Errors counter and the Datagrams Outbound No Route counter, both from the IP performance object, and the Connections Established counter from the TCP performance object. Explanations of the counters as provided by the MMC tool follow.

Performance Object	**Counter**	**Explanation**
IP	Datagrams Received Header Errors	Datagrams Received Header Errors is the number of input datagrams that were discarded due to errors in the IP headers, including bad checksums, version number mismatch, other format errors, time-to-live exceeded, errors discovered in processing their IP options, etc.
IP	Datagrams Outbound No Route	Datagrams Outbound No Route is the number of IP datagrams that were discarded because no route could be found to transmit them to their destination. This counter includes any packets counted in Datagrams Forwarded/sec that meet this `no route' criterion.
TCP	Connections Established	Connections Established is the number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.

Performance Object	Counter	Explanation
Memory	Pages/sec	Pages/sec is the rate at which pages are read from or written to disk to resolve hard page faults. This counter is a primary indicator of the kinds of faults that cause system-wide delays. It is the sum of Memory\\Pages Input/sec and Memory\\Pages Output/sec. It is counted in numbers of pages, so it can be compared to other counts of pages, such as Memory\\Page Faults/sec, without conversion. It includes pages retrieved to satisfy faults in the file system cache (usually requested by applications) non-cached mapped memory files.
Memory	Available MBytes	Available MBytes is the amount of physical memory available to processes running on the computer, in Megabytes, rather than bytes as reported in Memory\\Available Bytes. It is calculated by adding the amount of space on the Zeroed, Free, and Stand by memory lists. Free memory is ready for use; Zeroed memory are pages of memory filled with zeros to prevent later processes from seeing data used by a previous process; Standby memory is memory removed from a process' working set (its physical memory) on route to disk, but is still available to be recalled. This counter displays the last observed value only; it is not an average.
Physical Disk	Avg. Disk Queue Length	Avg. Disk Queue Length is the average number of both read and write requests that were queued for the selected disk during the sample interval.

Performance Object	Counter	Explanation
Processor	% Processor Time	% Processor Time is the percentage of elapsed time that the processor spends to execute a non-Idle thread. It is calculated by measuring the duration of the idle thread is active in the sample interval, and subtracting that time from interval duration. (Each processor has an idle thread that consumes cycles when no other threads are ready to run). This counter is the primary indicator of processor activity, and displays the average percentage of busy time observed during the sample interval. It is calculated by monitoring the time that the service is inactive and subtracting that value from 100%.

Performance tests were run on various weekends in September 2004 to ensure that COM Server and Client applications performed as expected. The set of performance objects and counters were chosen specifically to allow I2F testers to determine if problems such as memory leaks, processing bottlenecks, network transmission and or disk processing bottlenecks are evident.

An External System Simulator (ESS) application was developed specifically for the purpose of stress testing the COM Server by simulating up to 60 clients and three DSPs, the maximum number of system components external to the COM Server. In some of the performance runs, the ESS simulated 54 clients and ran in conjunction with 6 real clients to determine if the stress levels produced by the 54 simulated clients interfered with the normal operation of the 6 real clients. All 54 simulated clients exercised a simulated hardware PTT every second causing a steady stream of messages to be sent to the server. No degradation was noticed in the operation of the 6 real clients during these tests. Additionally, in weekend long performance test runs, all 60 simulated clients exercised a simulated hardware PTT every second causing a steady stream of messages to be sent to the server providing a much heavier load than any real life scenario. The performance tests were run for durations of 24, 48 and 72 hours and in each case, the review of the recorded performance run showed no anomalies in the performance object counters described above that were cause for concern.