

To the Graduate Council:

I am submitting herewith a thesis written by Eric Bernard Wrushen entitled "Automated Testing in Television Engineering". I have examined the final copy of this thesis for form and content and recommend that it is accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Daniel B. Koch

Daniel B. Koch, Major Professor

We have read this thesis
and recommend its acceptance:

M. J. Koch

Herbert B. Kupp

Accepted for the Council:

Curminkel

Vice Provost and Dean of

The Graduate School

Automated Testing in Television Engineering

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Eric Bernard Wrushen

August 1995

Copyright 1995 by Eric Bernard Wrushen
All rights reserved

ACKNOWLEDGEMENTS

This research was conducted at the University of Tennessee-Knoxville (UTK) during the period of March 1994 through August 1995. This project was sponsored by a consumer electronics company.

Many thanks to Dr. Daniel B. Koch, my major professor for his valuable time and advice. Your personal enthusiasm and interest is quite contagious. Being part of the ATF Team has been a tremendous experience and learning opportunity for me. It is obvious that you understand that the joy of having knowledge is realized fully when one freely gives it away.

Thanks also to Dr. Michael Roberts and Dr. Herbert P. Neff for their assistance in serving as my committee members.

Finally, to my lovely wife Victoria, thank you for all of your support, patience, and understanding. You have made it so easy for me to pursue and realize my goals.

ABSTRACT

This thesis is based on a research project involving the conversion of manual test procedures to automated ones. It investigates the motivating factors for automated testing, its benefits and disadvantages of implementing such a system. Since, the testing was to be executed on televisions, the fundamental theory of television operations is presented. Furthermore, the graphical code development environment as it applies to this research project is analyzed. Advantages and disadvantages of using LabVIEW with pioneering software engineering principles is also discussed.

The thesis then bridges theoretical principles with practical applications by escorting the reader through developed code. Instructions for the logic and functionality are also presented.

Finally, manual testing time results are compared to their automated counterparts to provide an assessment of time savings and ultimate cost reduction.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION-----	1
2. BASIC TELEVISION THEORY-----	8
The Color Signal -----	8
The Tuner -----	15
The Picture Tube -----	16
The Audio Section -----	18
3. AUTOMATIC TESTING FACILITY -----	21
Manual Testing vs. Automated Testing-----	22
The Physical Layout -----	28
The Interface Bus -----	30
System Equipment -----	34
System Control -----	35
Sun SPARCstation 10 -----	35
Switching and Routing -----	35
Tektronix TSI-8150-----	35
SVS 16/16 Router-----	37
Signal Generators -----	37
RE540 Stereo Generator-----	37
SMT02 Signal Generator -----	38
PM5640A Signal Generator -----	38
Staco 120 -----	38
Measurement Devices-----	39

	VM700A -----	39
	Tektronix 2714-----	39
	Magtrol Model 5100-----	39
	Miscellaneous -----	40
	SA 6350 Transmitters-----	40
	Variable Attenuators-----	40
	Summary -----	41
4.	SOFTWARE ENGINEERING WITH LABVIEW-----	42
	Structured Dataflow-----	43
	The Editing System -----	46
	LabVIEW with GPIB -----	49
	The Cleanroom Development Process -----	51
	Motivation -----	51
	Components of Cleanroom-----	52
	The Process -----	54
	System Requirements-----	54
	Specification -----	54
	Incremental Development -----	55
	Correctness Verification -----	56
	Quality Certification-----	58
	ATF Implementation of Cleanroom-----	59
	Motivation -----	59
	System Requirements-----	60
	Specification -----	61
	Design by Incremental Development----	62
	Verification -----	68

	ATF Certification-----	71
	Delivery -----	71
	Postmortems -----	72
	Documents-----	72
	User Interface Specification -----	72
	Cleanroom Process Description -----	73
	Summary-----	74
5.	ATF SOFTWARE-----	76
	RE540 -----	76
	Reinitc.vi -----	78
	R540errc.vi-----	80
	Re540c.vi-----	83
	SMT02 -----	91
	Smtinitc.vi-----	93
	Smterrc.vi -----	95
	Smt02c.vi-----	95
	RF Overload-----	103
	Test Panel-----	104
	Block Diagram-----	109
6.	CONCLUSION -----	122
	LIST OF REFERENCES -----	129
	VITA-----	132

LIST OF TABLES

TABLE	PAGE
3.1 Manual Testing Vs Automatic Testing-----	24
3.2 Typical RF Amplitude Levels -----	30
3.3 IEEE-488 Bus Pin Designations-----	32
3.4 Most Commonly Used Pins on an RS232 Bus-----	34

LIST OF FIGURES

FIGURE	PAGE
1.1 Diagram of the Five Basic Sections of a Television Receiver--	4
1.2 LabVIEW's Front Panel and Block Diagram -----	6
2.1 The 3.58 MHz Color-Reference Signal Located on Back-----	10
Porch of Horizontal Blanking Pulse	
2.2 Vector Relations of All Signals and Colors Used -----	11
in Televisions	
2.3 Fundamental Block Diagram of a Color Tri-Gun Display ----	12
2.4 Overall IF Bandwidth for Color Reception-----	14
2.5 Shadow Mask with In-Line Guns-----	17
2.6 Intercarrier Sound Takeoff Ahead of Detector -----	20
3.1 Manual Test System-----	22
3.2 Automatic Test System-----	23
3.3 ATF Principal Cost Areas-----	27
3.4 The Automatic Testing Facility-----	29
3.5 Block Diagram of Automatic Test Equipment Connections---	36
4.1 LabVIEW Structures-----	45
4.2 Sample Front Panel, Icon, and Connector Pane-----	47
4.3 Sample of a Block Diagram -----	48
4.4 The Cleanroom Software Engineering Process -----	53
4.5 Incremental Development Model of Prominent Life Cycle.--	57
4.6 Front Panel and Block Diagram of Black Box -----	62
4.7 Block Diagram of State Box -----	63

4.8	Block Diagram of Clear Box-----	64
4.9	ATF Incremental Development Plan -----	66
4.10	ATF LabVIEW Get Info Description -----	67
4.11	ATF's Correctness Verification Form -----	70
5.1	RE540 Stereo Signal Generator -----	77
5.2	Front Panel and Block Diagram for Reinitc VI -----	79
5.3	Front Panel of R540errc VI -----	81
5.4	Block Diagram for R540errc VI-----	82
5.5	Front Panel of Re540c VI-----	84
5.6	Block Diagram of Re540c VI-----	88
5.7	Rohde & Schwarz SMT02 Signal Generators-----	92
5.8	Front Panel and Block Diagram of Smtinitc VI-----	94
5.9	Front Panel of Smterrc VI-----	96
5.10	Block Diagram of Smterrc VI-----	97
5.11	Front Panel of Smt02c VI-----	99
5.12	Block Diagram of Smt02c VI -----	100
5.13	Block Diagram for Smt02c VI -----	101
5.14	RF Overload Skeletal Test Procedure -----	105
5.15	RF Overload Interconnectivity Diagram -----	106
5.16	RF Overload Front Panel -----	107
5.17	TV Header -----	110
5.18	RF Overload Block Diagram-Bit Stage-----	111
5.19	RF Overload Block Diagram-Initialization Stage-----	112
5.20	RF Overload Block Diagram-Preset Stage -----	114
5.21	RF Overload Block Diagram-Test Stage -----	116
5.22	View Panel for RF Overload Test Measurements -----	119

5.23	RF Overload Block Diagram-Reset Stage -----	121
6.1	Manual Test Times Vs Automatic Test Times-----	123
6.2	Composite of Automated Test Times-----	124
6.3	Average Test Times for Automatic & Manual Testing----	125
6.4	ATF Productivity Gains -----	127

Chapter 1

INTRODUCTION

The replacement of tedious and repetitive manual testing tasks with automation has helped many industries realize results that would otherwise have been unattainable. By automating these testing processes, businesses have seen sharp increases in consistency and productivity. Similarly, cost savings from the reduction in labor and rework have been realized. With manual testing, a highly skilled operator or technician is required to operate the equipment. These operations may include manual instrument control, instrument calibration, plug and cable changeovers, and precisely reading and logging measurement data [1]. On the other hand, with an automated test system, a computer is used as the controller. The implementation of an automated test system with a General Purpose Interface Bus (GPIB), allows the operator at a computer terminal to command various kinds of equipment to perform specific measurements of a unit under test.

Automated testing dates back to the period immediately following World War II. As test requirements became more complex, the demand for accurate, reliable, and consistent data increased dramatically too. This effort was led by the Aerospace Industry and the Federal Government. The government's automated testing interest centered around electronic warfare and providing a means to easily and reliably test parts, components, and

systems for the collection of thousands of data with varying parameters [2].

The automated testing concept spilled over to the area of Microelectronics and had a profound impact on that industry. Many standard Large and Very Large Scale Integrated (LSI, VLSI) circuits such as RAMs, ROMs, and microprocessors now undergo automated testing routines daily [2].

It is imperative that organizations be cognizant of the major cost areas that are associated with automated testing. Realizable cost saving benefits associated with automated testing over manual test methods would be lost if cost containment management is not carefully applied. This management style requires that the major cost areas, automated testing software (ATS) development, and automated testing hardware (ATH) installation and maintenance adhere closely to the project's schedule and budget.

This thesis details the steps, methodologies, and engineering principles used by a project team to develop automated testing software for the television industry. The project is divided into three phases over the span of one and one half years. The first phase of the project which is scheduled for completion December, 1995, is the focus area for this thesis. The principal requirement was to develop interactive software that would automate the tasks of sixteen test procedures. The software would serve as the operating system for a modern Automated Testing Facility (ATF). The first group of tests were categorized as *cover on* tests - tests that were conducted without disassembling the housing and

exposing the internal components of a television receiver. This group of tests allows the manufacturer to compare various characteristics of his television to that of his competitors.

The project team was assembled in April, 1994. The team consisted of four Electrical and Computer Engineering Master's Degree Candidates. To successfully write the programs, an understanding of televisions and the LabVIEW software package were necessary. Therefore, a course on basic television theory was included in the team's training schedule. Similarly, an interactive LabVIEW training series was completed by the team.

The primary function of modern televisions is to receive radio frequency (RF) signals that are comprised of audio and video information. Once received, these signals are converted to sound and light patterns for listening and viewing purposes.

The television has five main parts that interact to produce the sound and picture display. The chassis detects and processes the desired channel's signal that is forwarded to it from the tuner. The chassis then sends the audio information and video signals to the speakers and picture tube respectively. The picture tube is where the picture is displayed. A yoke helps to direct the electron beam as it traces the display picture on the face plate of the picture tube. The sound is projected out through the speakers, synchronously with the picture display [3]. A block diagram of the main parts is shown in Figure 1.1.

A high-performance graphical programming language called LabVIEW from National Instruments was used to develop the

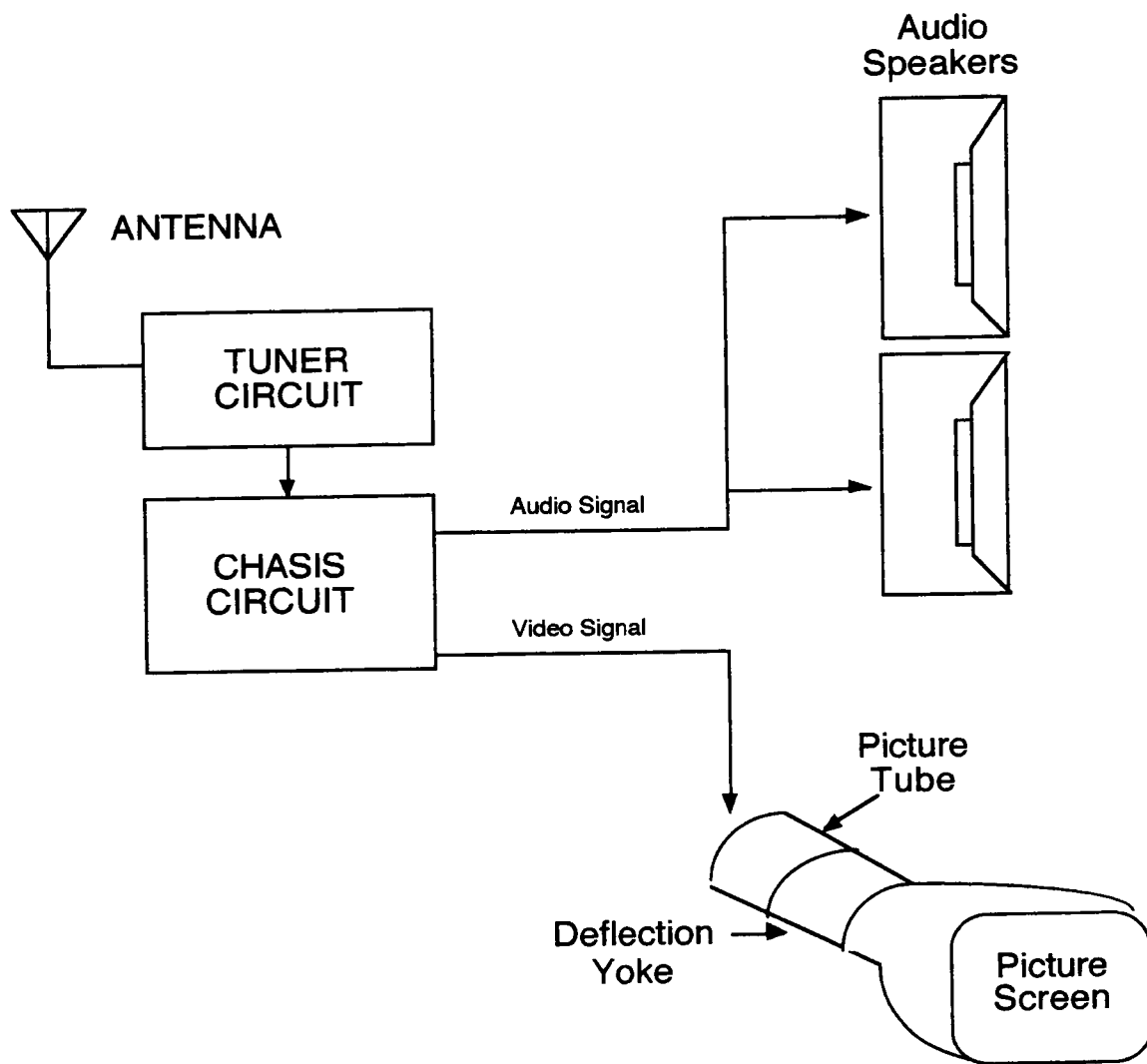


Figure 1.1 Diagram of the Five Basic Sections of a Television Receiver

instrument drivers and the automated testing software. LabVIEW is well suited for instrument control, data acquisition, analysis, and display. This language supports hierarchical modules, termed virtual instruments (VIs). Each VI includes a user interface panel and block diagram as shown in Figure 1.2. The VI behavior is defined by the connected icons on the diagram while the user interactively operates the VI via the front panel controls and indicators. The generous text editing environment allows for the addition of descriptive labels to define icons' functionality and operating environment. The culmination of all the aforementioned items along with the built-in GPIB icons made this package ideal for this project.

In an effort to employ software engineering principles throughout the project, the ATF Team utilized the Cleanroom Software Engineering Process. This process, while more meticulous than the popular design-and-debug software design approach, offered a systematic way to achieve zero-defect code. This was accomplished through rigorous system requirements defining, code verification, and certification testing. The Cleanroom Process and its elements as they apply to this project will be explored in a later chapter.

As an outline to the material presented following this overview, in the second chapter, the basic theory of televisions will be discussed. The third chapter will thoroughly discuss the Automated Testing Facility and its supporting hardware.

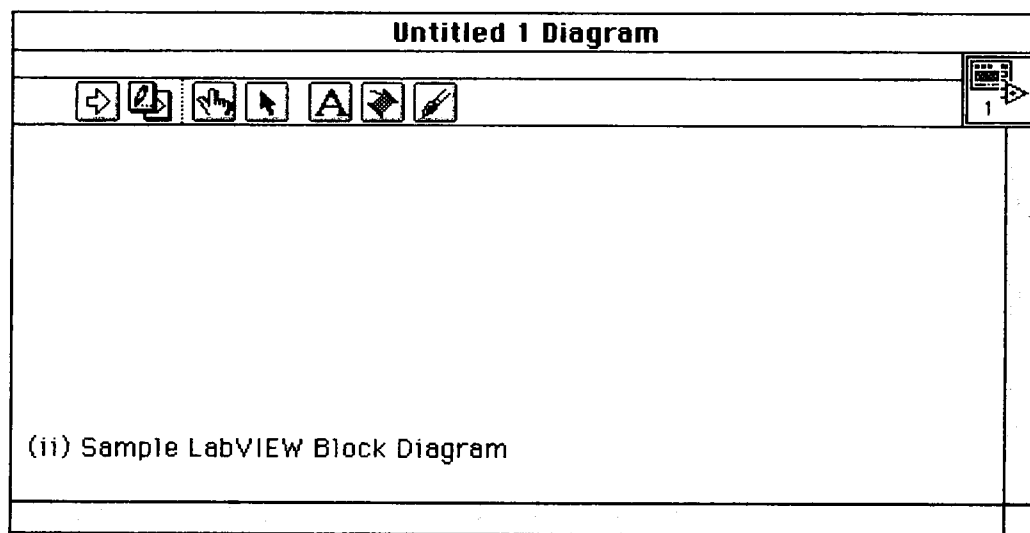
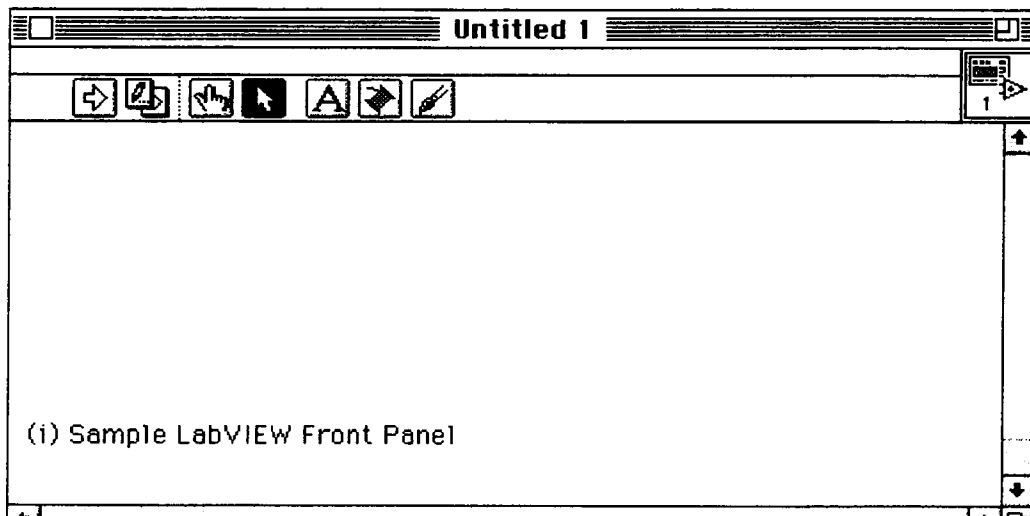


Figure 1.2 LabVIEW's Front Panel and Block Diagram

Furthermore, a description of the physical testing areas will be provided. The fourth chapter will elaborate on the use of the Cleanroom Process with LabVIEW. This chapter will discuss the conventions and adoptions of the Cleanroom Process for this graphical language. The fifth chapter will discuss in detail the software that was developed by the author. Finally, the conclusion will summarize results, give a synopsis of the project's success, and identify improvement opportunities for further development.

Chapter 2

BASIC TELEVISION THEORY

A brief description of the functions of the five main component areas of a television receiver was provided earlier; however, before embarking on a discussion of methods for testing televisions, a more thorough discussion of televisions is presented. The focal point of this discussion will be the fundamental theory of color receivers.

The Color Signal

There are three signals that make up the RF carrier modulation signal of color televisions. These are the luminance signal, the suppressed-carrier color signal, and the audio signal. The NTSC-FCC standardized color signal has the following composition specification:

$$E_m = E_y + [E_q \sin(\omega t + 33^\circ) + E_i \cos(\omega t + 33^\circ)] \quad (2.1a)$$

$$E'_q = 0.41(E_b - E_y) + 0.48(E_r - E_y) \quad (2.1b)$$

$$E_i = -0.27(E_b - E_y) + 0.74(E_r - E_y) \quad (2.1c)$$

$$Y = 0.30R + 0.59G + 0.11B \quad (2.1d)$$

The luminance signal Y is the sum of the signals $.30R$, $.59G$, and $.11B$, where the R , G , and B denote equal primary color voltages in reference to white light. Since the color signal must be transmitted as a separate signal, the color signal is generated by the modulation

of the picture carrier by a 3.579 MHz subcarrier [3]. An interfering 'beat' or heterodyne is created on the desired 3.58 MHz frequency; hence, this subcarrier is subsequently modulated by a color signal and suppressed to alleviate the heterodyne. The process generates sidebands for transmission that extend from 2.5 MHz to 4.3 MHz to yield a null-to-null bandwidth of 1.8 MHz. The resultant color subcarrier is formed by adding the two subcarrier frequency signals. The in-phase component I , and the quadrature component Q , are defined as

$$I = -0.60R + 0.28G + 0.32B \quad (2.2a)$$

$$Q = 0.21R - 0.52G + 0.31B \quad (2.2b)$$

Interestingly, the amplitude of each subcarrier becomes zero for white, that is when red (R) equals green (G) and green equals blue (B). By adding the luminance and chrominance signals in the receiver, the blue, green and red video channel voltages are derived as given in Equation set 3 [4].

$$B = Y - 1.10I + 1.70Q \quad (2.3a)$$

$$G = Y - 0.28I + 0.65Q \quad (2.3b)$$

$$R = Y + 0.96I + 0.62Q \quad (2.3c)$$

As seen in Figure 2.1, the chrominance signal has associated with it a colorburst which is a phase-reference component located on the "back porch" of each horizontal blanking interval. The 3.58 MHz phase reference component or 'burst' keeps the carrier in phase

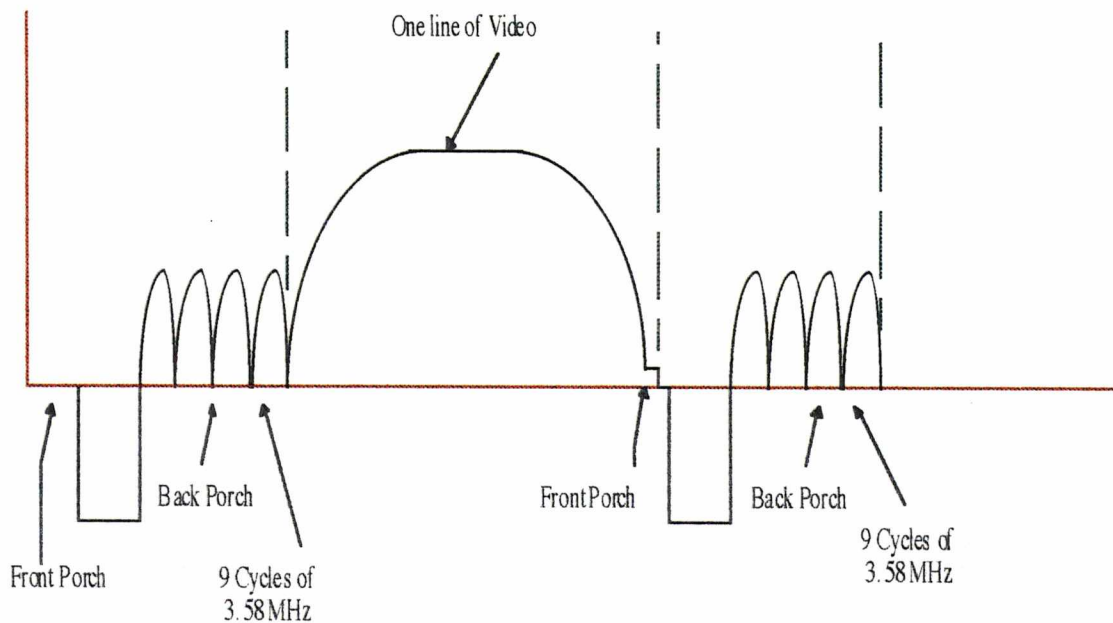


Figure 2.1 The 3.58 MHz Color-Reference Signal Located on Back Porch of Horizontal Blanking Pulse

with the transmitting signal; thus, allowing the carrier to be regenerated at the television receiver. The color burst consists of nine cycles of the subcarrier frequency.

The color burst, which is a vector display of each color's phase relation to I, Q, and the transmitted reference is shown in Figure 2.2. Note the I and -I vectors correspond to the Q sidebands. Only one I and one Q signal are transmitted at any given time. All colors in the burst can be represented through phase and amplitude modulation of I and Q. Hence, by altering the vector lengths

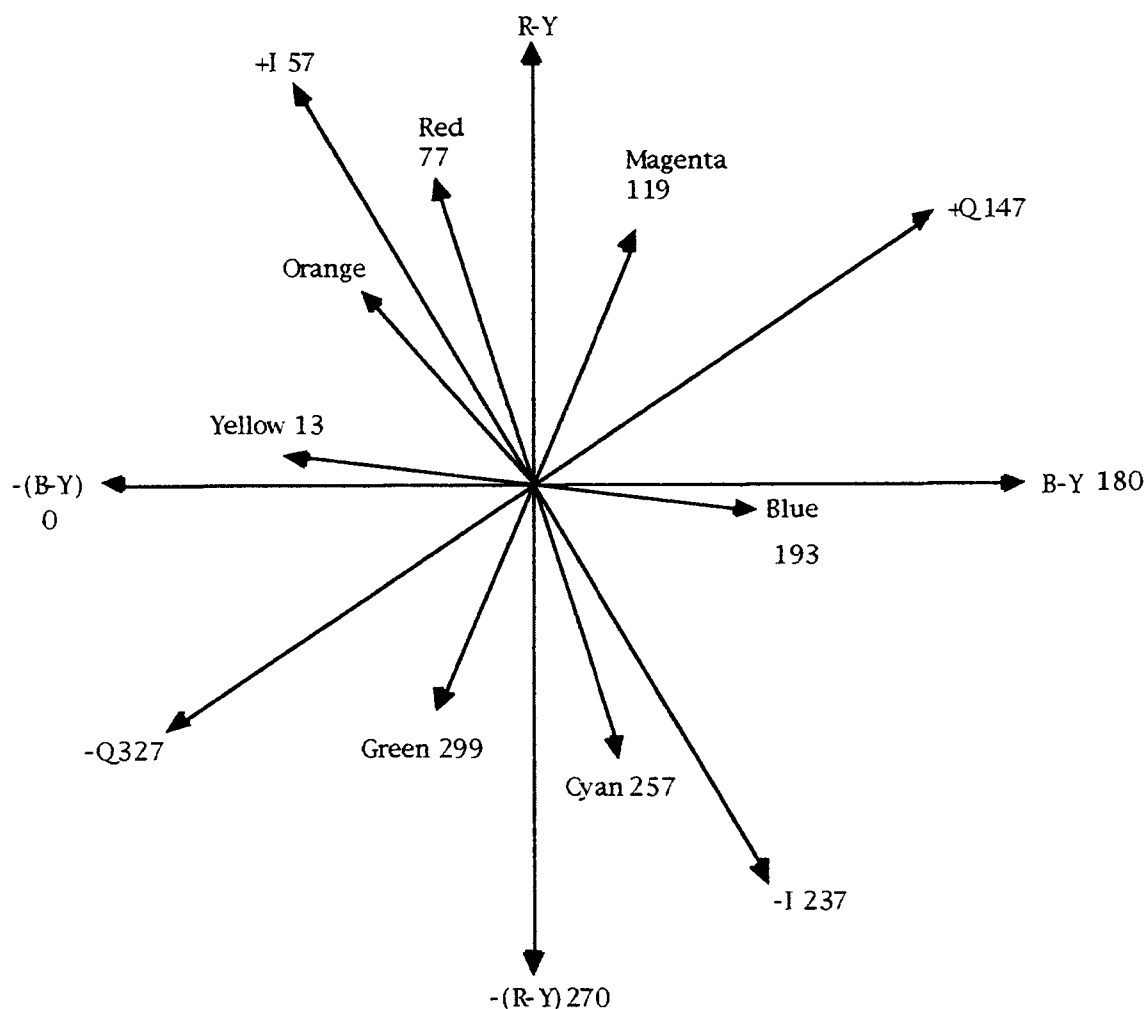


Figure 2.2 Vector Relations of All Signals and Colors Used in Televisions

(sideband energies) and shifting in/out of phase (+,- sidebands), all colors may be regenerated 3.58 million times per second [4].

The basic block diagram of a television receiver that uses a simultaneous three-gun picture tube display is shown in Figure 2.3. The parts of the outline that are valid only in color receivers are shown with heavy outline. The dashed lines represent an optional

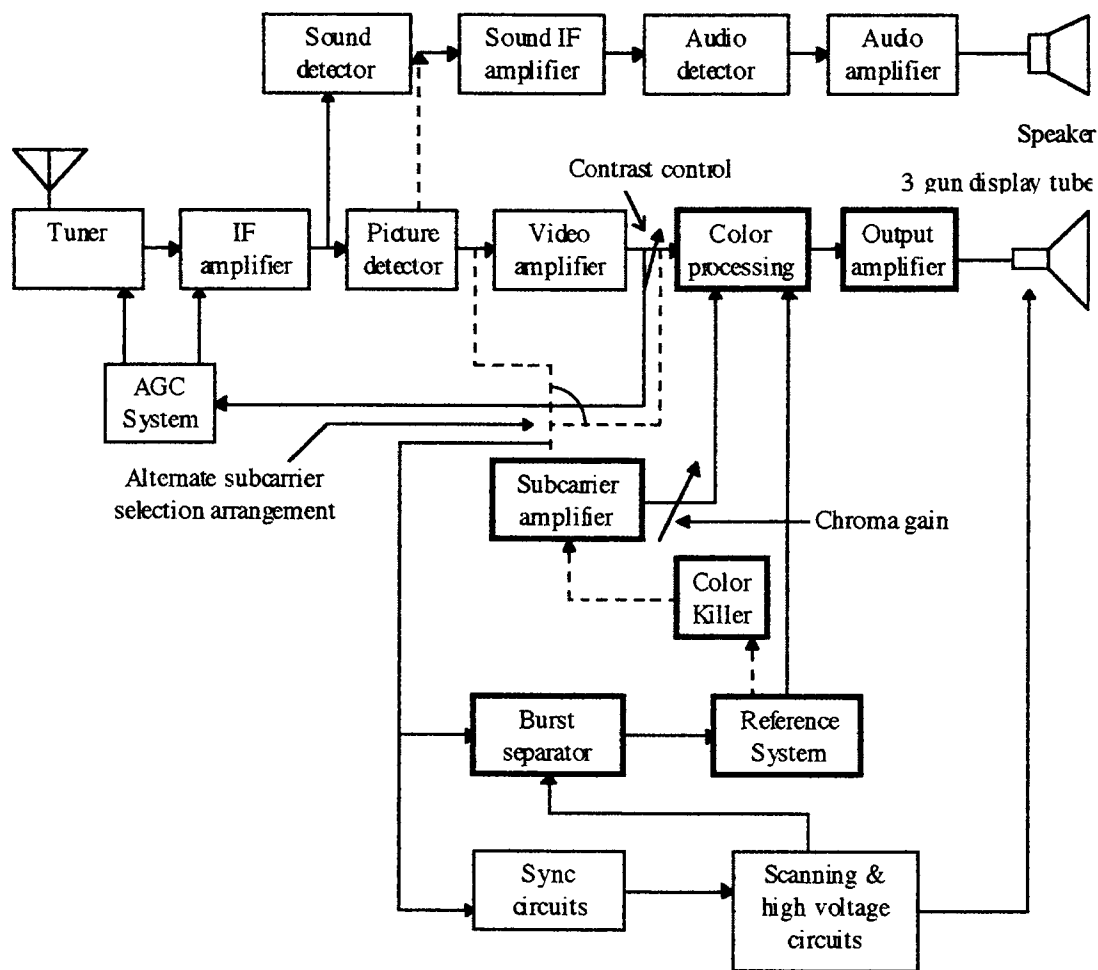


Figure 2.3 Fundamental Block Diagram of a Color Tri-Gun Display

design for the color tri-gun display. Blocks that are common to monochrome and color receivers are shown with the lighter outline.

In the color bandpass amplifier section , the composite video signal is sent to the color bandpass amplifier for separation of the color subcarrier and color burst from the video signal. There is a gain control on the amplifier that behaves as a chroma controller for

the adjustment of the amount of luminance and color signal in the display.

The burst separator's primary function is to select and amplify the color burst signal from the chroma subcarrier signal (burst separator). After the amplification process, the local reference oscillator is phased locked to the color burst signal. The two signals have identical frequencies.

The color processor utilizes synchronous detectors to demodulate the color subcarrier. This results in either the I and Q or R-Y and B-Y color difference signals. The reference circuit provides carrier drive of reference phase for the demodulator circuitry. For accurate demodulation, precise control of the reference phase is a must for proper color fidelity. The third color difference signal G-Y is created by combining the R-Y and B-Y signals. The output amplifier supplies the necessary gain (20-30 times) to have enough amplitude to drive the three-gun Cathode Ray Tube.

In order to attain correct color signal reception, the IF bandwidth must accommodate the color subcarrier modulation sidebands. These sidebands extend to a frequency of 41.5 MHz as depicted in Figure 2.4 below. The IF amplitude versus frequency response must be stable and constant as the input signal level varies if a uniform level of color saturation is to be sustained. Note that the color subcarrier is 3.58 MHz below the picture carrier and is modulated 'in phase' and 'in quadrature' to produce two sets of

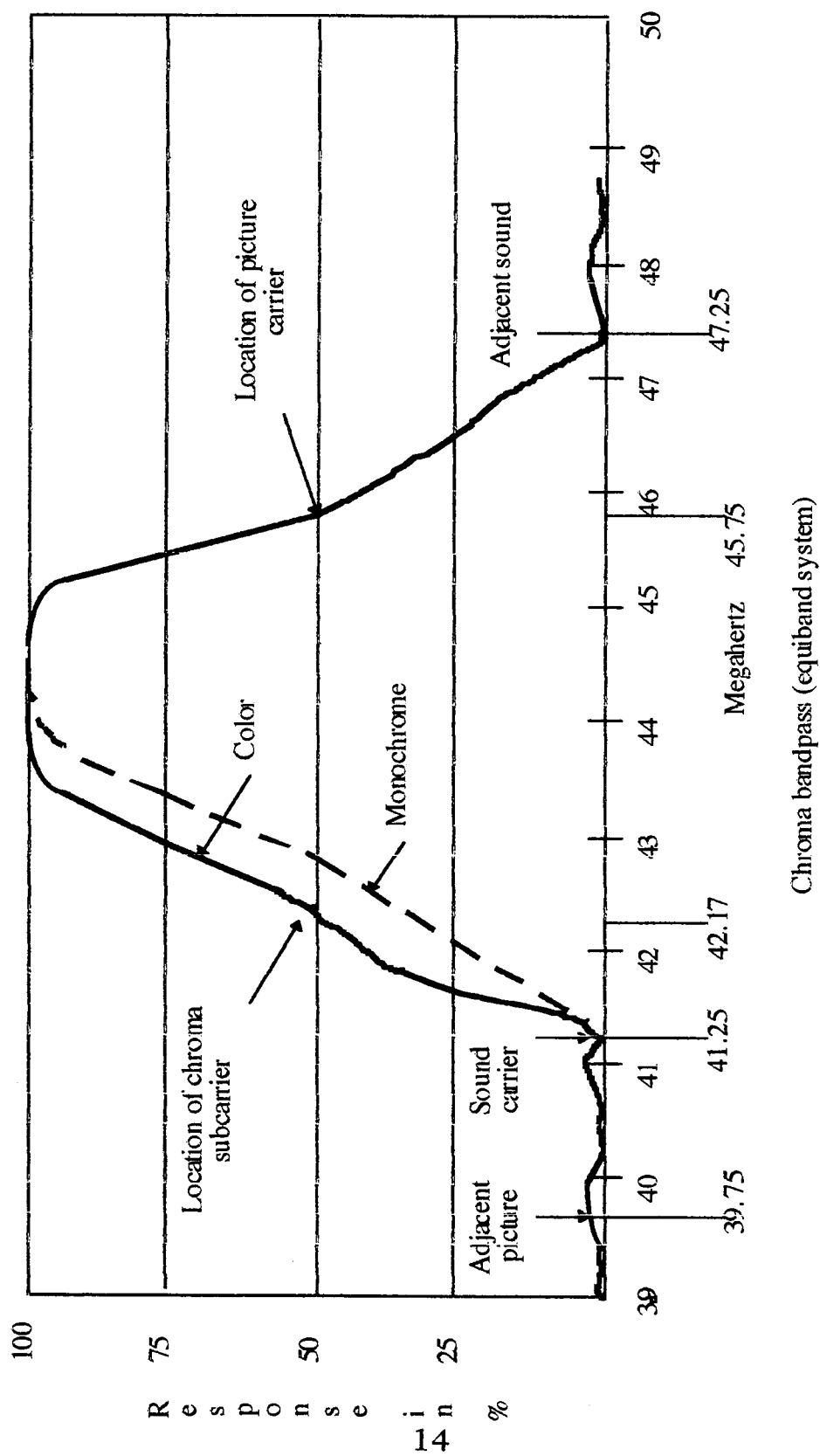


Figure 2.4 Overall IF Bandwidth for Color Reception

sidebands. The lower frequency (I) sideband extends 1.5 MHz below the color subcarrier [3].

The Tuner

The purpose of tuners is to select the desired signal available at the antenna and to exclude all others. The IF amplifier amplifies the signal to a level that allows demodulation of the information that is located on the carrier, while maintaining the lowest possible noise level. Each channel allocated for transmission of a television signal encompasses 6 MHz of RF spectrum. The tuner usually provides a 10 MHz passband and a gain of 15 to 20 dB for the channels at the 3 db points.

The passband is generally attained by four tuned circuits: a tuned preselector located after the antenna and before the RF stage, a double tuned interstage network between the RF and mixer stages, and a single tuned coupling circuit at the output of the mixer [3]. Most VHF tuners have a high-pass filter and trap section located at the tuner input to allow signals above 54 MHz to proceed to the tuner and reject signals that are below 54 MHz (channel 2). This section plays a major role in interference reduction that could be produced by nearby transceivers, and short-wave transmitters [4].

The UHF tuner contains a preselective tunable filter followed by a diode mixer. The tunable local oscillator fundamental operates at 45.75 MHz above the selected UHF picture carrier; thus,

producing an IF output signal having a picture carrier of 45.75 MHz and a 41.25 MHz sound carrier.

The mixer stage accepts as inputs the RF signal and the output of the tunable local oscillator to yield the intermediate frequency of 45.75 MHz for the picture carrier. The local oscillator operates at a frequency which is always 45.75 MHz above the incoming picture frequency. These frequencies were initially chosen such that the interference of one television receiver with another would be minimized by always having the local oscillator fundamental above each VHF television band. Since the UHF band is continuous in the 470-890 MHz band, with no gaps between channel groups, the FCC-UHF channel allocation plan avoids assigning transmitters with a 15 channel separation in the same geographical area [4].

The Picture Tube

Since all secondary colors may be created from proper mixture of the three primary colors, three electron guns are required in a color picture tube. Three light-emitting phosphors are deposited on the picture-tube screen with each respective electron gun providing excitation according to incoming video signals. Two of the most common constructs for the shadow mask are the “round hole” and the “vertical slot” apertures [4]. For round holes, the screen is as shown in Figure 2.5. An aperture mask is used to ensure that each beam strikes only its respective color phosphors. The three electron beams emanating from the electron guns

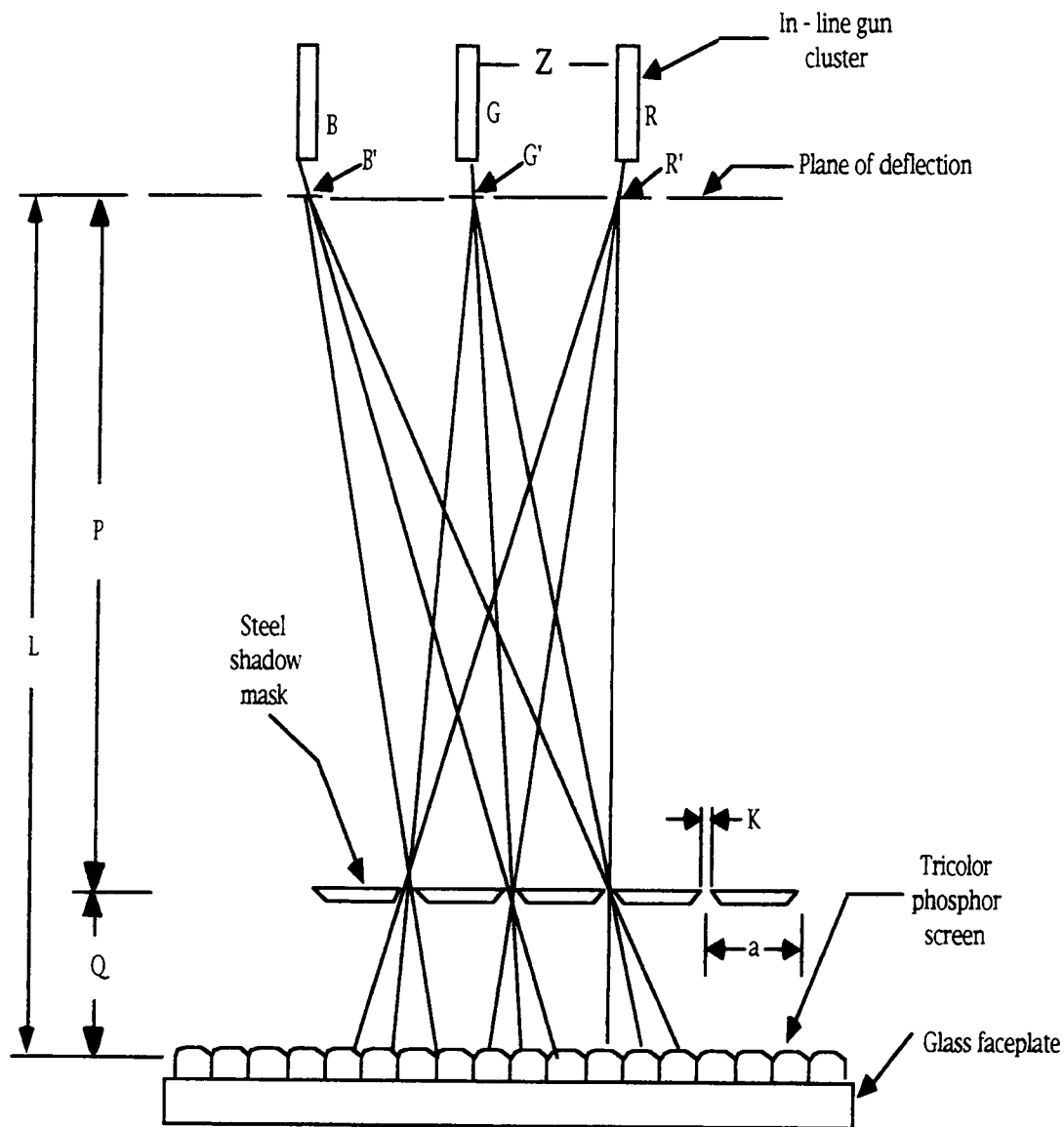


Figure 2.5 Shadow Mask with In-Line Guns

converge on holes in the aperture mask, then diverge precisely to strike their respective color dots. The beams may overlap more than one hole and the holes are encountered only as they happen to fall in the scan line. The aperture mask provides a “shadow” on the phosphors; hence, preventing accidental excitation of phosphors by incorrect electron beams [3].

In Figure 2.5, the points B', G', and R', lying in the deflection plane, represent apparent centers of deflection for blue, green, and red beams striking an aperture of interest. It is noted that the deflection centers move with varying deflection angles. The extension of a ray from the centers of deflection to the faceplate marks the printing spot for each respective color of the tricolor group. The centers of deflection are spaced a distance Z apart. This distance Z directly affects the dot spacing on the phosphor screen. The projection of Z goes as the ratio Q/P as depicted in Figure 2.5. The diagram further shows how the mask hole horizontal pitch 'a' projects as screen horizontal pitch in the ratio L/P . This ratio is also applicable to the vertical pitch. The mask-to-panel spacing (Q) is chosen such that the dot size is optimized with no overlapping. This leads to the conclusion that equally spaced dots is the desired screen geometry [3].

The Audio Section

Sound transmission is an FM signal having a maximum deviation of ± 25 KHz (100% modulation). The signal provides an audio bandwidth of 50-15,000 Hz. The 4.5 MHz difference between video and audio (sound carrier is transmitted at a frequency 4.5 MHz above the RF picture carrier) carriers is the sound IF signal [4].

The intercarrier sound system passes the IF picture and sound carriers (45.75 and 41.25 MHz, respectively) through a detector to create the intermodulation beat of 4.5 MHz. The intercarrier sound signal is subsequently amplified, band limited, and FM demodulated

to retrieve the audio signal. The block diagram of an intercarrier sound system is shown in Figure 2.6. In a discrete system, the intercarrier detector is often a diode detector feeding a 4.5 MHz resonant circuit. The sound and picture IF signals are carried all the way to the video detector in an IC IF system. Under this construction, one output port of the balanced synchronous demodulator supplies both the 4.5 MHz sound carrier and the composite baseband video signal.

The volume control is usually located between the sound detector output and the audio amplifier input. The audio amplifier and speaker in a television set are almost identical to those found in FM radios. The audio amplifier contained in a sound system IC can provide output power in the 2-5 W range [4].

Speaker impedances are usually 16 Ω . Sufficient audio gain is provided to drive full-rated output with a 30 percent modulated input signal. In general, there are no unusual requirements for audio sections in TV receivers [3].

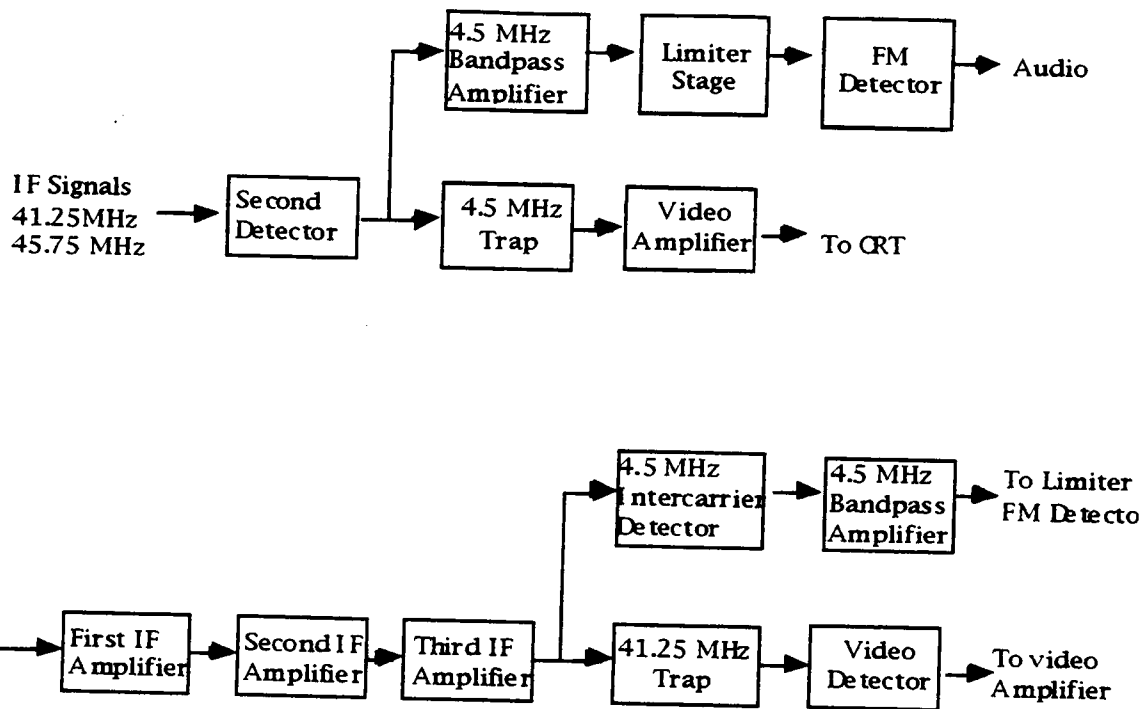


Figure 2.6 Intercarrier Sound Takeoff Ahead of Detector

Chapter 3

AUTOMATIC TESTING FACILITY

As previously mentioned, the evolution of automated testing had its beginning in the late 1940's. With the increasing availability and decreasing cost of the mini-computer, the employment of workstations in an automated testing facility has increased greatly. The lower cost, increased capability and reliability of these machines have helped to make such an undertaking easier to justify economically.

The two types of tests that were of interest for this project were the Built In Test (BIT) and the Unit Test. The first stage of BIT consisted of the Power-Up Diagnostic routine which provided 'Go/No-Go' information for each device in the test system. The second stage checked measurements of specific interconnectivity paths and the functionality of individual equipment or sub-systems of the Automatic Testing Facility (ATF). This phase entailed using generators to produce patterns and measurement equipment to report results. The measurements were then stored in a loss table if tolerances were not exceeded. The third stage was used to identify calibration errors [6].

Unit tests utilize various test devices to form a test system. These test systems unlike BIT must be connected to a Unit(s) Under Test (UUT) for proper functionality [5]. Unit tests are the focal point of this discussion.

Manual Testing vs. Automated Testing

Manual test systems are characterized as test facilities where each test depends on separate human actions to select, connect and control testing devices, observe, interpret and record measurements made and evaluate test data. Note that Figure 3.1 shows the typical components of a manual test system [5].

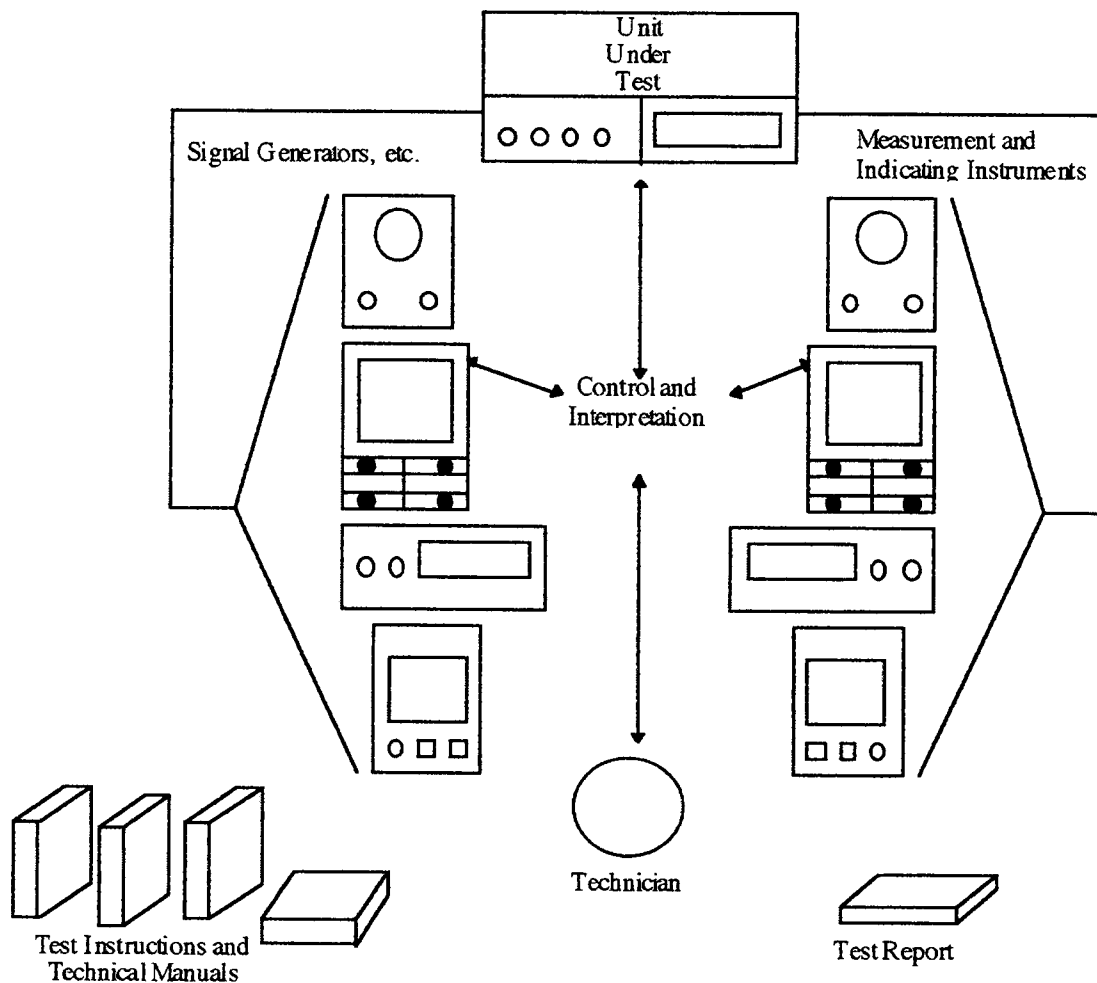


Figure 3.1 Manual Test System

On the other hand, automatic test systems require programming test functions prior to testing, such that the functions are performed automatically during the test procedure through the use of automatic test equipment (ATE). Figure 3.2 show the major elements that are usually found in an automatic test system [2].

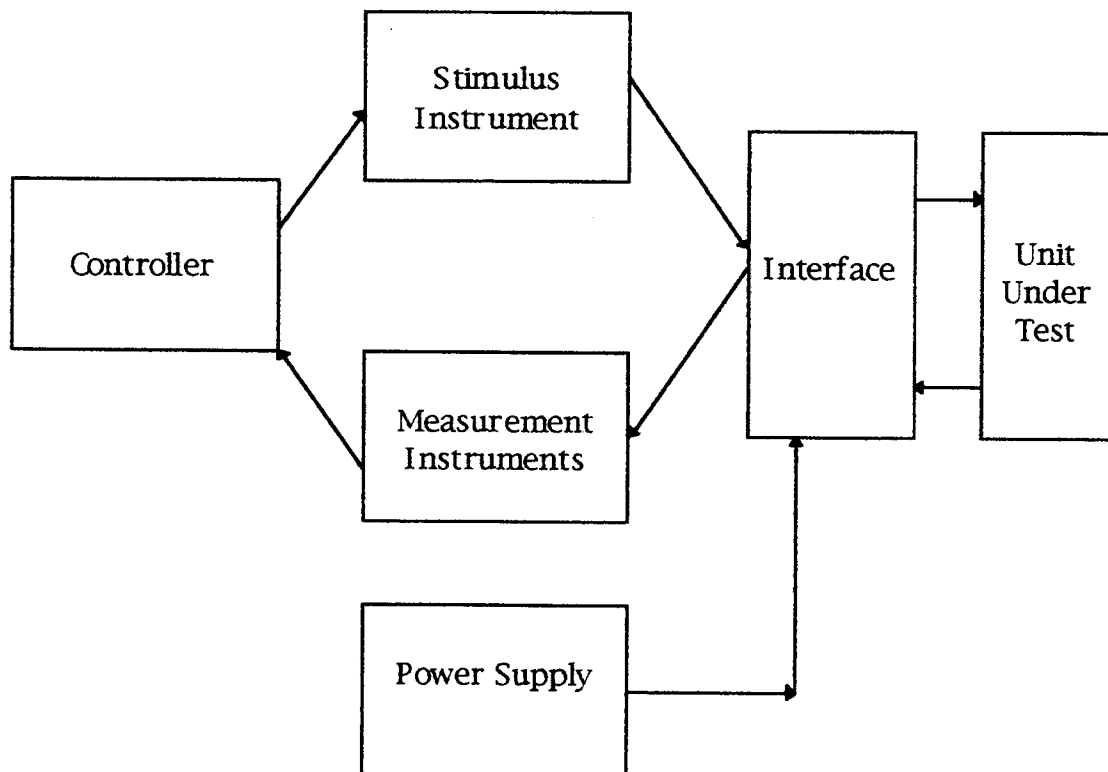


Figure 3.2 Automatic Test System

In a manual test cycle, the operator is involved in a number of discrete actions. Though the procedures vary from situation to situation, in general, the functions listed in Table 3.1 are included.

This differs greatly from the automatic test cycle where the test procedures depend heavily on the detailed content of the test program and the ATE's sophistication. Compare the normal procedure for an operator of an ATE to that for manual operations in Table 3.1.

Table 3.1 Manual Testing Vs Automatic Testing

	Manual Test Cycle	Automatic Test Cycle
1	Arrange for the availability of UUT	Arrange for the availability of UUT, ATE, and test program
2	Read the test schedule and test instructions	Connect the UUT to the ATE
3	Assemble the necessary items of test equipment, jigs, leads, etc.	Apply the test program and start the ATE
4	Initiate the relevant report blanks	Observe the test results that fail to meet program requirements
5	Connect the various items of test equipment to the UUT	Rectify the fault
6	Apply and adjust test inputs, power signals, controls, etc.	Repeat functions 4, 5, & 6
7	Select display/meter scale, observe trace reading. Record Results	Disconnect the UUT from the ATE
8	Compare measured results to nominal values & tolerances	Remove and process the test print-out
9	Interpret meaning of compared values & decide necessary action	
10	Repeat functions 7, 8, 9, & 10	
11	Disconnect the test equipment	
12	Complete the test report	

A straightforward appraisal and evaluation of ATF costs would consist of defining the expected life, cost and functionality of other alternatives, establishing the required profit and anticipated

cost reductions during the life of alternative systems, and comparing equipment costs with the labor content of operation.

For simple test systems, the aforementioned methods may be adequate; however, these factors prove to be inadequate for most systems. Many attributes of an automated test system cannot be unambiguously measured in financial terms alone. Although, it is fairly easy to create a list of expected benefits and disadvantages, it is often nearly impossible to attach meaningful cash values to these headings, particularly to the benefits [5].

The cost of providing a central control and processing facility represents a fixed sum for a given control capacity, and varies in steps for different capacities. The sizes of the steps tend to decrease as the capacity increases. This is largely due to the better use of input and output devices and control interfaces. Naturally, the added memory storage and signal switching requirements cause a parallel increase in cost.

The complexity and range of the UUT that is to be tested affect the capital cost dramatically. The cost of peripheral hardware and software are minimized for a single UUT or similar UUT. Hardware, software, and BIT costs increase as an automated test system becomes comprehensive and multipurpose. Often, the growth in the number and type of UUT causes a shift to a comprehensive system [5].

With manual testing, the bulk of operating cost is that incurred in payment for the operator's labor. Assuming that labor

is employed effectively, the cost can be expressed in terms of a cash rate per man-hour. Figure 3.3 shows the major ATF cost areas.

On the other hand, the cost of operating an ATF is more difficult to express and hence to translate into an equivalent cash rate per hour. This is because the cost depends greatly on the success of the user organization in planning and implementing effective ATE utilization. Factors that must be considered are listed below [2, 5]:

1. The proportion of total time when the ATE can be used.
2. The time taken by the ATE for self-test routines
3. The periods of interruption to which the ATE is exposed (e.g., time for changing UUT's, human intervention in the test procedure, rectification of failures during the test cycle)
4. The periods when productive work flow or availability of UUT's is interrupted for maintenance testing.
5. The utilization of the ATE for activities other than its primary testing role

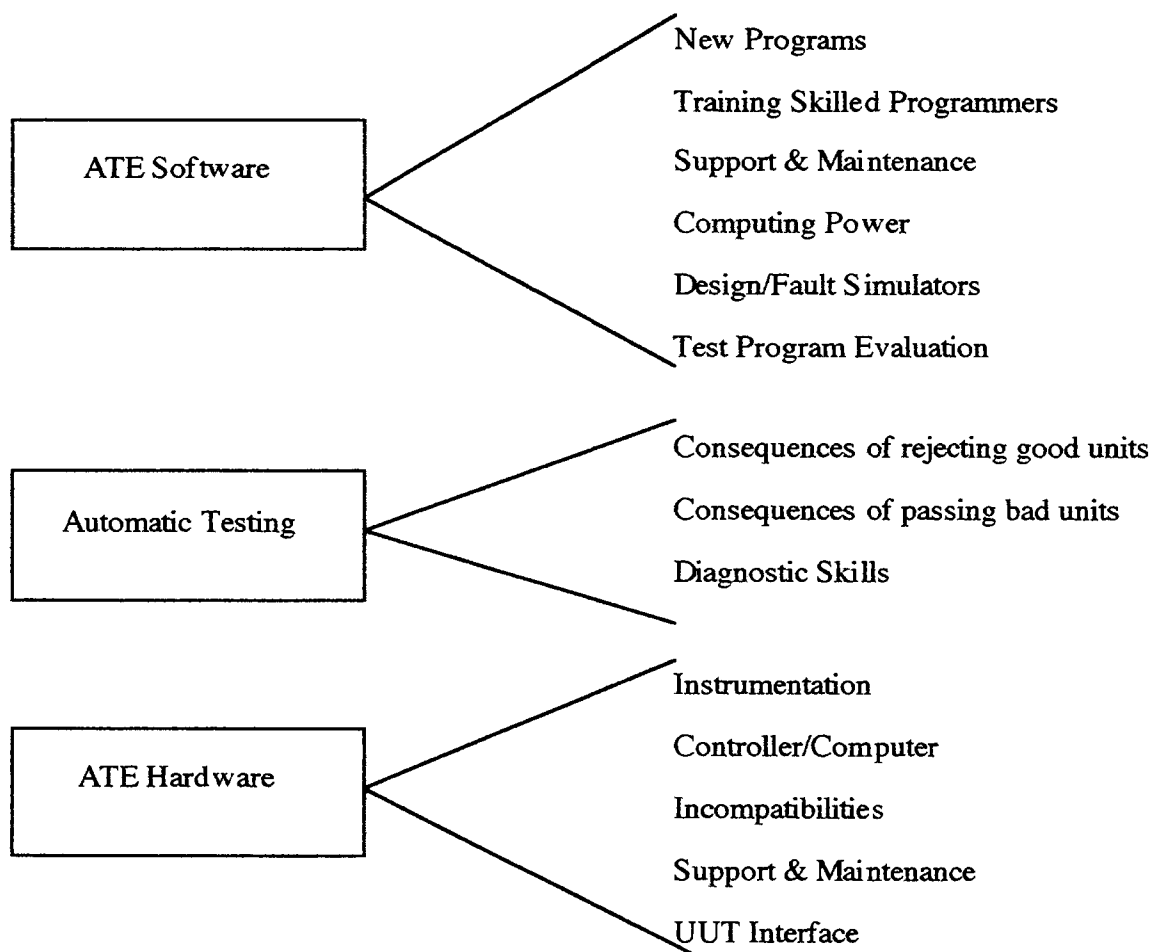


Figure 3.3 ATF Principal Cost Areas

The Physical Layout

The ATF associated with this project is comprehensive in its design. The physical layout of the ATF is given in Figure 3.4. The focal point of the Automated Testing Facility is the ATF Equipment Room. This room houses the three racks of equipment that are used for measuring and monitoring various parameters of the units under test as well as performing some system diagnostics. The main control center, the Sun SPARCstation 10, is also housed in this room. The Sun SPARCstation controls all equipment via an IEEE-488 General Purpose Interface Bus (GPIB). There is also an observation station here. This station is where televisions under test reside if the testing is done in this room.

The ATF Equipment Room is adjacent to the Demonstration Room. There is an interior window between the two rooms. The Demonstration Room is used to show different characteristics and performance levels of varying models of televisions under test to outside parties that may be interested in such data. The interior window is directly in front of the workstation (control center) in the ATF Equipment Room. Hence, this layout allows an individual to execute a test at the X-terminal, control all equipment needed for a test, and test parameters. The Demonstration Room's perimeter is lined with observation stations; however, the operator must choose a station that is visible from the control center and to the viewing audience [6].

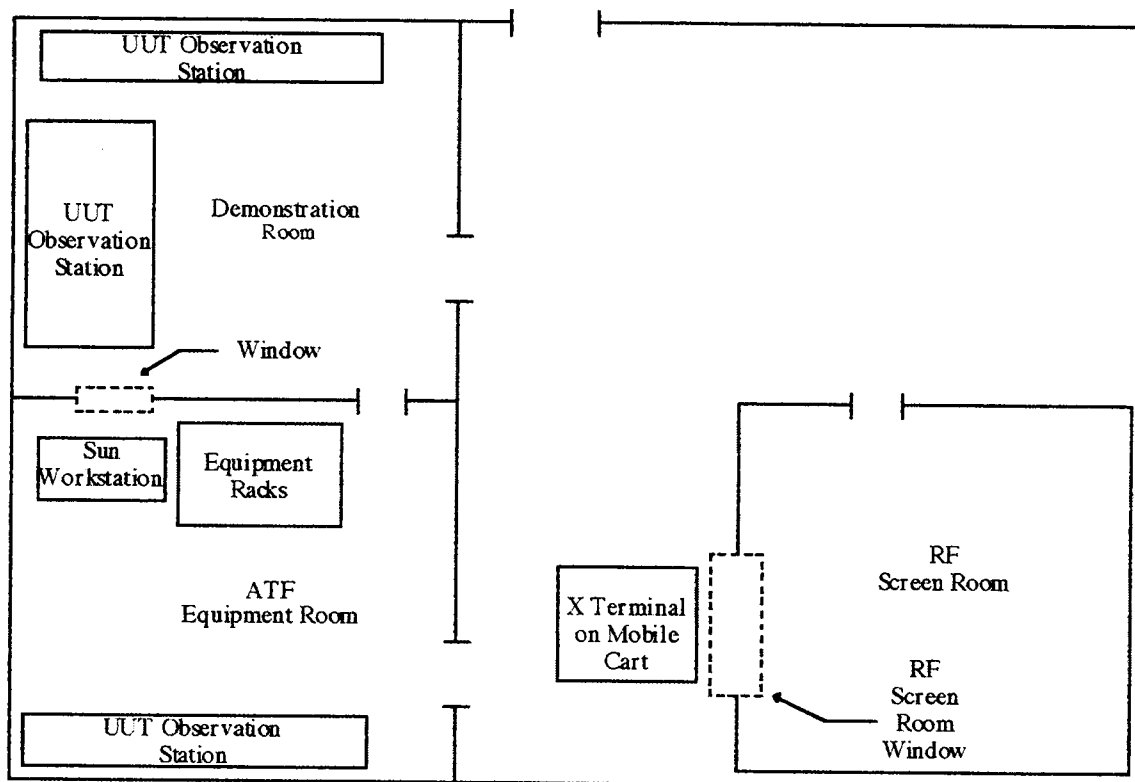


Figure 3.4 The Automatic Testing Facility

The Radio Frequency (RF) Screen Room offers the operator an environment free of spurious RF signals. This room was needed because certain tests that either measure RF levels or require that the RF level be within a specific tolerance could not be accurately ascertained otherwise. Obviously, the RF Screen Room can not keep all spurious signals out; however, the room filters most of the undesirable RF signals by suppressing them with approximately 120 dB of attenuation. Most tests' RF signal levels range from ten microvolts to one volt. The various levels are described in Table 3.2 [6].

Table 3.2 Typical RF Amplitude Levels

Characteristic	RF Level
Sensitivity	10 - 20 microvolts
Useable	50 microvolts - 100 millivolts
Standard Cable	1 millivolt - 3 millivolts
Overload	100 millivolts - 1 volt

Due to the remote location of the Demonstration Room and the RF Screen Room relative to the test equipment racks, a mobile cart was used for operations in these areas. This cart is connected to the stationary part of the system via a GPIB bus extender and coaxial cable. This mini subsystem has a remote X-terminal with a standard Unix keyboard and mouse as input devices. Located also on this remote station are several measurement and observation devices [6].

The Interface Bus

The Automated Testing Facility utilizes the IEEE-488 bus to interconnect all of its equipment and the controller. This bus was introduced by the Hewlett Packard Corporation in 1978, where it was called the HP-IB. Because of the high data transfer rates of up to one megabyte per second, the bus quickly gained popularity in other applications. The versatility of the system prompted the name General Purpose Interface Bus (GPIB). The bus allows up to 15 instruments to be connected together. Each instrument can be a talker or a listener. Each device terminates the bus with a 3 k Ω

pull-up, and 6.2 k Ω pull-down resistor. One device load should be connected every two meters, yet the overall length must not exceed twenty meters. Device loads must not be separated by more than 4m. An extender is placed between the host controller and the rest of the bus and is electrically invisible to the system. Its function is to provide low impedance driving power to the bus to handle the additional capacitance and inductance of bus cabling when more than the recommended 2-meter distance between devices is required. Bus expanders are used in the event more than fourteen instruments need to be interconnected [7].

IEEE-488 instruments usually have a connector for interfacing to the bus, and switches that are used for setting the address. Some instruments have secondary addresses that give access to different parts of the instrument [7]. Secondary addressing is a provision in the interface system to allow multiple addressing, which permits the controller to go one step further into the instrument to control a specific portion of it. When addressing a GPIB controlled device with a secondary address, the controller command must contain both address codes [8].

Implementation of the IEEE-488 bus requires sixteen active lines. The lines are classed into three groups, as shown in Table 3.3. The transfer bus consists of three lines that control the transfer of data on the data bus by ensuring a fixed sequence of events. It checks that handshaking between devices occur. It also ensures that the last data byte was accepted before allowing the

transmission of a new data byte. The DAV line is set by the talker, after it has placed data on the lines. The listener sets NDAC during the process of reading this data, or NRFD if it is unable to accept data at that instance.

Table 3.3 IEEE-488 Bus Pin Designations

Designation	Pin Number	Description
Data Bus	1	data bus
	2	data bus
	3	data bus
	4	data bus
	13	data bus
	14	data bus
	15	data bus
	16	data bus
Control Bus	5	end or identify (EOI)
	9	interface clear (IFC)
	10	service request (SRQ)
	11	attention (ATN)
	12	shield
	17	remote enable (REN)
Transfer Bus	6	data valid (DAV)
	7	not ready for data (NRFD)
	8	not data accepted (NDAC)
Grounds	18	ground
	19	ground
	20	ground
	21	ground
	22	ground
	23	ground
	24	logic ground

The control bus consists of five lines. When the ATN line is activated by a controller it indicates to all the instruments, on the bus, that they must give up use of the bus and interpret the data bus as commands. The IFC line is used to initialize instruments on the bus. The SRQ line is used by an instrument to signal to the controller that it requires attention. The REN line is used by the controller to set an instrument in a 'programming accept' mode. The EOI is used to indicate the end of a multi-byte message by a talker, or with ATN this executes a polling sequence. Alternatively, a line feed character can be sent to indicate an end of a message [7].

The Automated Testing Facility consists of two GPIB's. Each bus originates at the Sun workstation. Bus zero interconnects most of the system's equipment. This bus uses secondary addressing to communicate with some of its equipment. Bus one is the home of the equipment that will be used on the cart for remote testing. This bus uses both a bus extender and secondary addressing for communicating. Recall that the bus extender is needed to accommodate lengthy bus lines, namely those greater than 20 meters.

The system also uses several IEEE-488.2/RS232 converter boxes to convert from the parallel GPIB to a serial communication link. This was needed since several of the system's devices were serial instruments. The RS232 bus was developed in 1962 to connect computers to peripherals. The RS232 bus uses a 25 line

connector. Table 3.4 shows the designation of the pins that are used most. Using the example of a computer and X-terminal communicating the following description will fully describe the functions of the RS232 line. The TDATA and RDATA lines handle data communication to and from an X-terminal respectively. The RTS line warns the controller to prepare to receive data. The CTS line receives the acknowledgment from the controller that it is ready to receive data. The DSR line is for the controller to communicate with the X-terminal, to let it know that it is operational. Similarly, the DTR line allows the X-terminal to tell the controller that it is operational [7].

Table 3.4 Most Commonly Used Pins on an RS232 Bus

Pin Number	Description
1	protective ground to chassis
2	data line (TDATA)
3	data line return (RDATA)
4	request to send (RTS)
5	clear to send (CTS)
6	data set ready (DSR)
7	signal ground
20	data terminal ready (DTR)

System Equipment

The first phase of the ATF project utilizes 3 racks of highly sophisticated equipment. These range from measurement instruments, video generators, audio generators, and transmitters to name a few. A discussion of the various devices that make-up the

Automated Testing Facility follows. The instruments have been classified into 5 categories: System Control, Switching and Routing, Signal Generators, Measurement devices, and Miscellaneous. A detail of the system equipment and interconnectivity of control and data lines are shown in the block diagram of Figure 3.5 [6].

System Control

Sun SPARCstation 10

The intelligence and capability for system control are embedded in the Sun SPARCstation 10. This mini-computer has a one gigabyte hard drive for data storage. It is also outfitted with a three and one half inch diskette drive and a compact disk read only memory drive (CD ROM). A large twenty-one inch color monitor was chosen for the unit. The SPARCstation is a Small Computer System Interface (SCSI) device; hence, for successful control of the GPIB lines, it needed a SCSI to GPIB converter unit as an interface [9].

The SPARCstation also is a host computer in a small local area network (LAN) that supports a Tektronix X-terminal via an Ethernet connection for remote operations [10].

Switching and Routing

Tektronix TSI-8150

The Tektronix TSI-8150 test system interface is a multi-function interface unit. The unit is a chassis that provides power

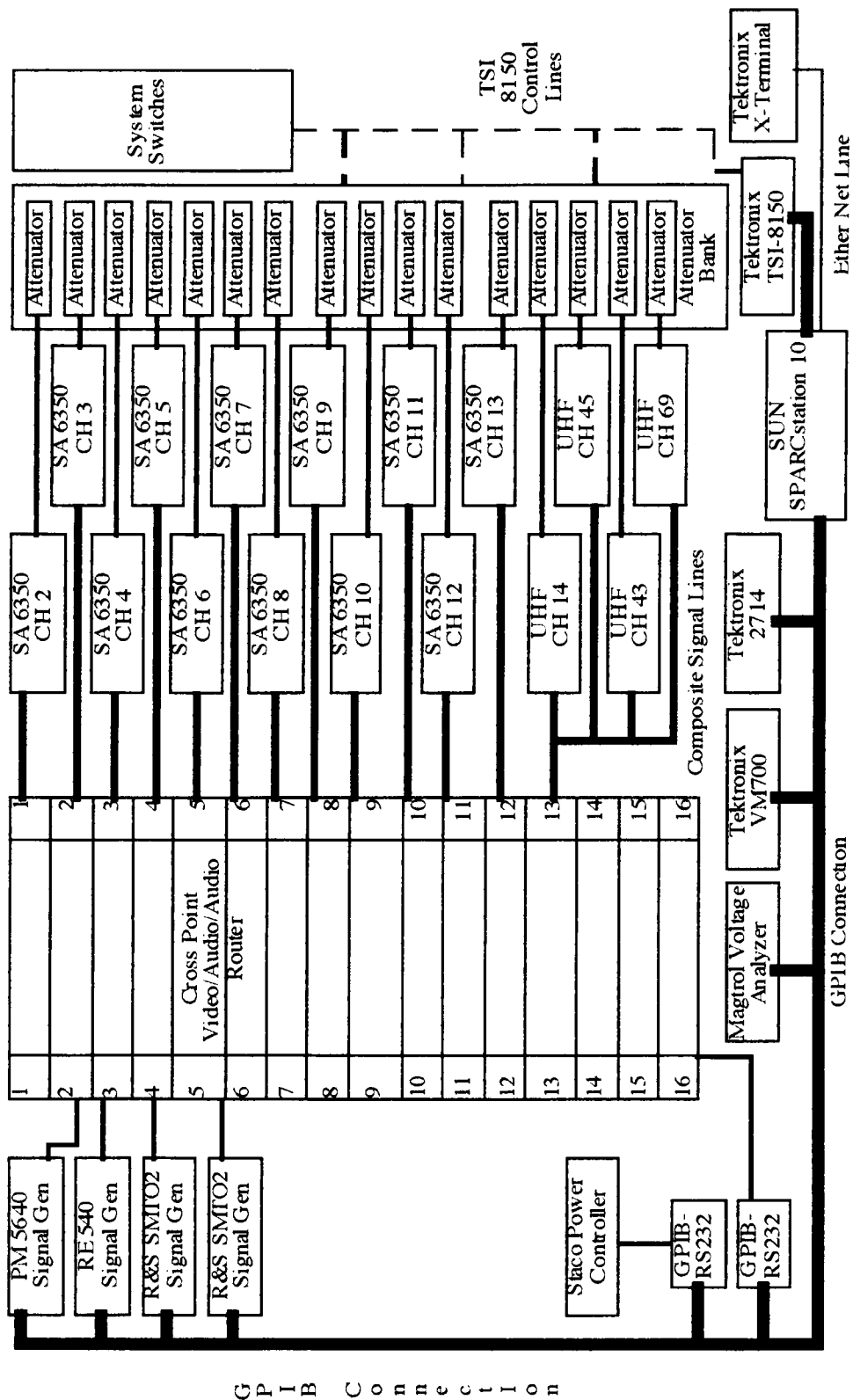


Figure 3.5 Block Diagram of Automatic Test Equipment Connections

and interface facilities for a number of modules and cards. The TSI-8150 is the system switching controller, providing this necessary function for hardware configuration of automated testing. The TSI-8150's addressing scheme utilized the rear of five slots. Slots R3 through R6 were used to control the system's attenuators. Slot R1 was used entirely for the relay switches of the system. A thorough discussion of the TSI-8150 occurs in Chapter 5 [11].

SVS 16/16 Router

The Sierra Video System Router is a routing switcher for audio signals, video sync, and machine control. There are sixteen inputs and sixteen outputs designated for signal cross-routing. For the Automated Testing Facility, the router is configured with three levels at its input and output terminals. Two levels are designated for audio and one is for video [12].

Signal Generators

RE540 Stereo Generator

The RE540 TV Stereo generator is designed for development and test of audio circuits in TV sets that conform to the (NTSC) specifications. The RE540 contains digital, synthesized multi-tone modulation generators, that supply a composite stereo signal with pre-emphasized (L+R) and DBX encoded (L-R) signals. The RE540 offers all stereo modes (L=R, L=-R, R only, L only, and R&L). The modulation signal covers the frequency range of 50 Hz to 15 KHz with a resolution of 50 Hz and an equivalent input level range up to

99% with 0.1% resolution. For an input range of 100% to 150%, the RE540 has a 1% resolution [13].

SMT02 Signal Generator

The SMT02 Signal Generator is a highly flexible instrument that provides a number of modulation modes. It has a specified lower frequency limit of 5 KHz. Its upper frequency limit is 1.5 GHz. The SMOT2 is capable of amplitude modulation, frequency modulation, phase modulation, and pulse modulation [14].

PM 5640A Signal Generator

The PM 5640A NTSC video test signal generator produces a large variety of color and black & white signals. The instrument produces five classes of signals. These are square waves, pulse and bar line waves, linearity, color bar and black & white signals [15].

STACO 120

The STACO 120 voltage controller is a microprocessor based controller designed for controlling motor driven variable transformers. The controller accepts set point and feedback voltage inputs and provides output signals until the feedback voltage matches the set point. The input signal may be a voltage, current, or computer interface type signal. The feedback signal may be one of several voltage levels or frequencies. Input signals are measured using peak, average, or RMS techniques. Multiple set points may be programmed [16].

Measurement Devices

VM700A

The VM700A Video Measurement set is a multi-function television test and measurement device. It performs the functions of a waveform monitor, vectorscope, automatic measurement set, and a noise measurement set on acquired television signals [17].

Tektronix 2714

The 2714 Spectrum Analyzer has incorporated in its logic the capability to measure, display, and monitor many frequency domain characteristics. These characteristics can be viewed in a variety of modes as well. These include determination of signal maximum and minimum, measuring signal-to-noise ratios, measuring signal bandwidth and amplitude levels [18].

Magtrol Model 5100

The Magtrol 5100 Single Phase Power Analyzer measures load voltage for AC and DC circuits with range maximums of 15, 30, 150, 300, and 600 volts rms. The 5100 measures and monitors current for DC and AC circuits for ranges of 2, 5, 10, 20, and 50 amps and RMS respectively. The power is computed and displayed in watts with a resolution of ± 0.01 percent of active volts and amps range [19].

Miscellaneous

SA 6350 Transmitters

The Model 6350 Television Modulator generates a high quality vestigial sideband TV signal. The circuit produces excellent picture and sound fidelity from color or black and white video inputs and from either baseband audio or 4.5 MHz subcarrier inputs. The modulator uses a Surface Acoustic Wave (SAW) filter that provides superior adjacent channel performance. The video modulator module amplitude modulates (AM) a 45.75 MHz continuous wave (CW) carrier with baseband video (1V p-p) to generate the Video IF. The Broadband Audio Modulator (BAM) module processes baseband (monaural and NTSC composite) and 4.5 MHz external subcarrier signal to generate the sound IF signal centered at 41.25 MHz [20].

Variable Attenuators

The system uses computer controlled variable attenuators to control RF signal levels. These attenuators are passive in nature and allow suppression in one decibel steps. The attenuators have eight pads that allow a range of control from 0 dB when all pads are not engaged to 127 dB when all pads are invoked. The values at the pads are 1, 2, 4, 8, 16, 32, 32, and 32 dB. The TSI-8150 controls the switches that allow the pads of the desired attenuator to be invoked.

Summary

To summarize findings, automated testing has many advantages over manual methods. In most cases, ATE's run tests faster, are more consistent, experience fewer errors and cost less on a per test basis. The utilization of an ATF frees the highly skilled technician to perform work more suited for his skill level and allows testing to be done by a far less skilled and usually less expensive operator.

The disadvantages of implementing an automated testing facility are few, yet profound. On the average, they cost more than manual test sets to initiate. Properly utilized ATF's however, often pay for themselves in a few years. Due largely in part to the complex mix of hardware and software, reliability sometimes is a problem in an ATF. It is important that the integrity of the system is upheld so that the ATE cycles through the tests as they should. The advantages of automated testing outweigh the disadvantages; hence, it is expected that the need for ATE's will increase [21].

The next chapter will elaborate on the use of the Cleanroom Process with LabVIEW. This chapter will discuss the conventions and adoptions of the Cleanroom Process for LabVIEW's graphical interface.

Chapter 4

SOFTWARE ENGINEERING WITH LABVIEW

One quest of the software development team was to apply engineering design principles to the development of the code. It was believed that the application of a logical and methodical approach would yield more robust code, in less time than industry's current design-and-debug method. The team chose The Cleanroom Software Development Process (Cleanroom) as a roadmap. While the process was developed with textual languages in mind, it was found that many elements of Cleanroom were well suited and quite adaptable to the graphical environment that was being used for the code development. The elements of "pure" Cleanroom and the closely simulated process adopted for this project will be discussed in this chapter; however, the tool used for the code development will be discussed first.

The programming language of choice for this project was National Instruments' LabVIEW version 3.1. LabVIEW is a high performance graphical programming system. LabVIEW is available on three computer platforms: IBM PC's and compatibles, Apple Macintoshes, and Sun SPARCSTATIONS. Its uses range from analysis, data acquisition and simulation to instrument control. It consists of a graphical editor for the visual programming language, G, a compiler, and a multithreaded execution environment. LabVIEW is fundamentally object-oriented by design [22]. LabVIEW designs

consist of a hierarchy of modules, called virtual instruments (VIs). These are comprised of a user interface panel (front panel), a block diagram, and an icon with connectors. The VI is interactively operated by a user through the graphical controls and indicators on the front panel. The behavior of the VI is determined by the block diagram, which is a structured data-driven dataflow graph of subVI icons and low level functions [23].

In a pure dataflow system, tokens (data) originate from the source terminals and flow along the wires until they come to a node. Once all of the node's input wires have tokens, the node 'fires' and computes output values that are subsequently propagated along the output wires. Cycles are disallowed in a pure dataflow system because the firing rule requires all inputs to have tokens prior to allowing a node to fire. The model lacks the ability to handle conditional or iterative operations.

Structured Dataflow

To prevent the problems associated with relaxing the firing rule and priming certain wires with tokens to accomplish conditional and iterative operations, LabVIEW utilized structured programming techniques. Namely, programming structures were built in as fundamental components, hence allowing LabVIEW to retain the clarity of strict firing rules and adding cyclic and conditional functionality.

Figure 4.1 shows the four control structures of LabVIEW. Each of the structures behaves like a node to the rest of the system. Thus, LabVIEW created structures that have internal conditional or cyclic behavior and external behavior as a node. The *while loop* structure has a counter and boolean continuation flag. After executing internal operations (flag is false), the while loop produces a token that is output through tunnels located at the loop's boundary; therefore, propagation of the token is possible by attaching an external wire to the tunnel.

The *for loop's* similarity to the while loop diverges only in the manner in which the loop terminates. Unlike the while loop, the for loop is executed a predetermined number of times. The number of loop iterations is dictated by the number that is wired to the integer sink N. Both the while loop and the for loop have an indexing option at each tunnel that is created at the structures' boundaries. This allows the loops to terminate when all elements of an array have been through its' respective iteration.

Both of the loop structures can have shift registers at the left side and the right side. The shift register consists of a left terminal and a right terminal. The left terminal is a sink to the outside, but a source to the loop body. Conversely, the right terminal is a sink to the loop and a source to logic external to the loop. A token supplied to the left terminal prior to execution initializes the variable. The token is emitted from the right terminal after execution of the loop. It will have the value of the variable of the previous iteration.

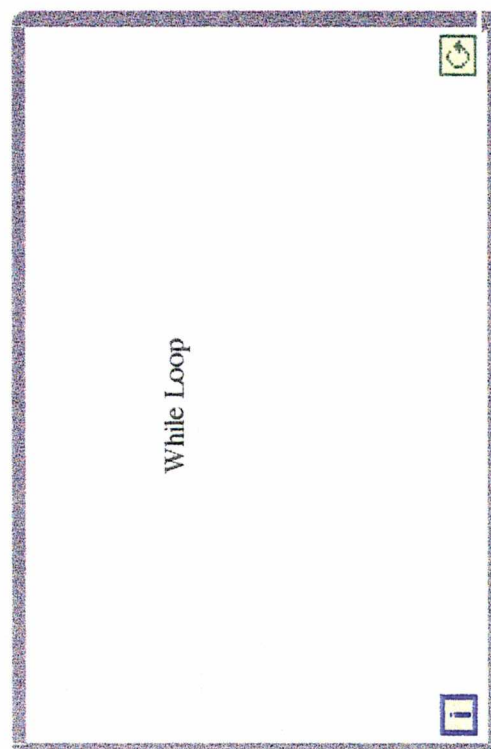
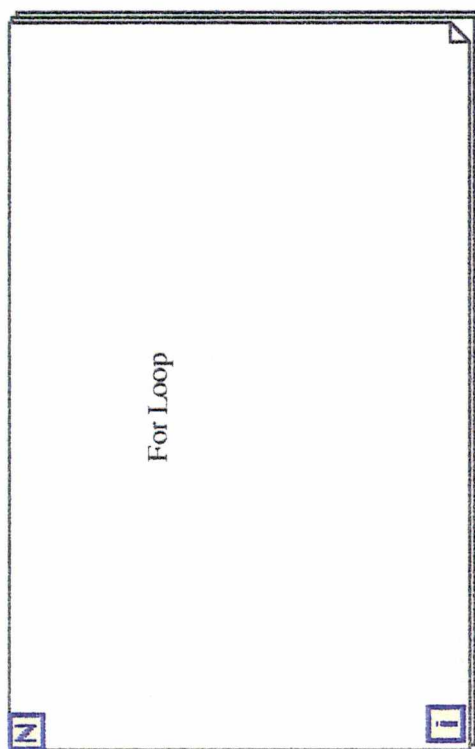
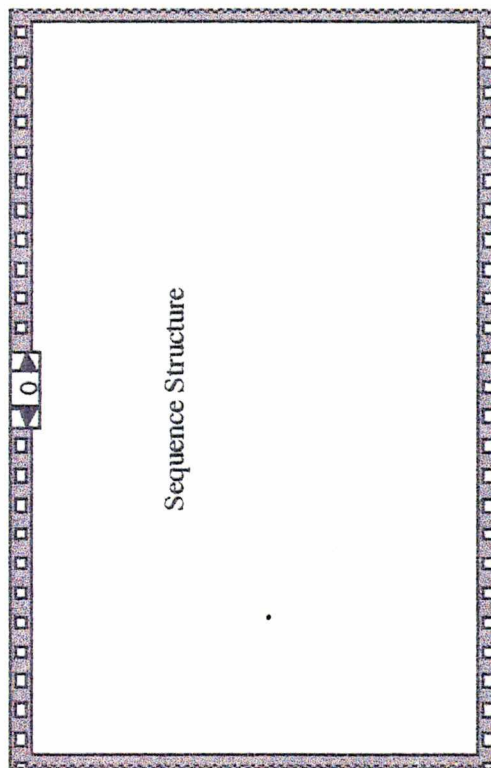
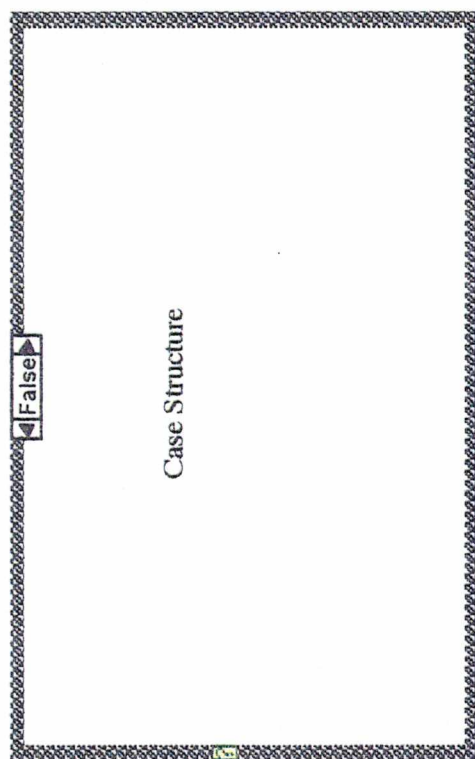


Figure 4.1 LabVIEW Structures

Multiple terminals are allowed for the left shift register. Each lower terminal is updated from the terminal above it prior to the start of the next iteration.

The *case* structure has a selector terminal that allows conditional executions. The selector terminal accepts either a boolean or a numeric token. The boolean instance has two overlaid cases that execute accordingly when a respective true or false token appears at the selector. With a numeric input at the selector, the case structure executes the case number corresponding to the value of the numeric. In both events tokens are exited through tunnels that have to be wired for each case.

Similar to the case structure, the *sequence* structure has multiple subdiagrams; however, all subdiagrams are executed in sequence. Its main use is to specify order of execution [23].

The Editing System

Each LabVIEW VI consists of a front panel, block diagram, and an icon/connector pane as pictured in Figures 4.2 and 4.3. VIs are constructed by selecting controls and indicators from seemingly endless menus of dials, knobs, switches, lights, meters, charts, and others. Controls are selected according to data type and desired display style. Controls and indicators that are located on the front panel have associated terminals located on the block diagram. The controls and indicators are wired to various functions and structures to form the VI. There are many functions to access.

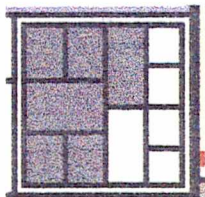
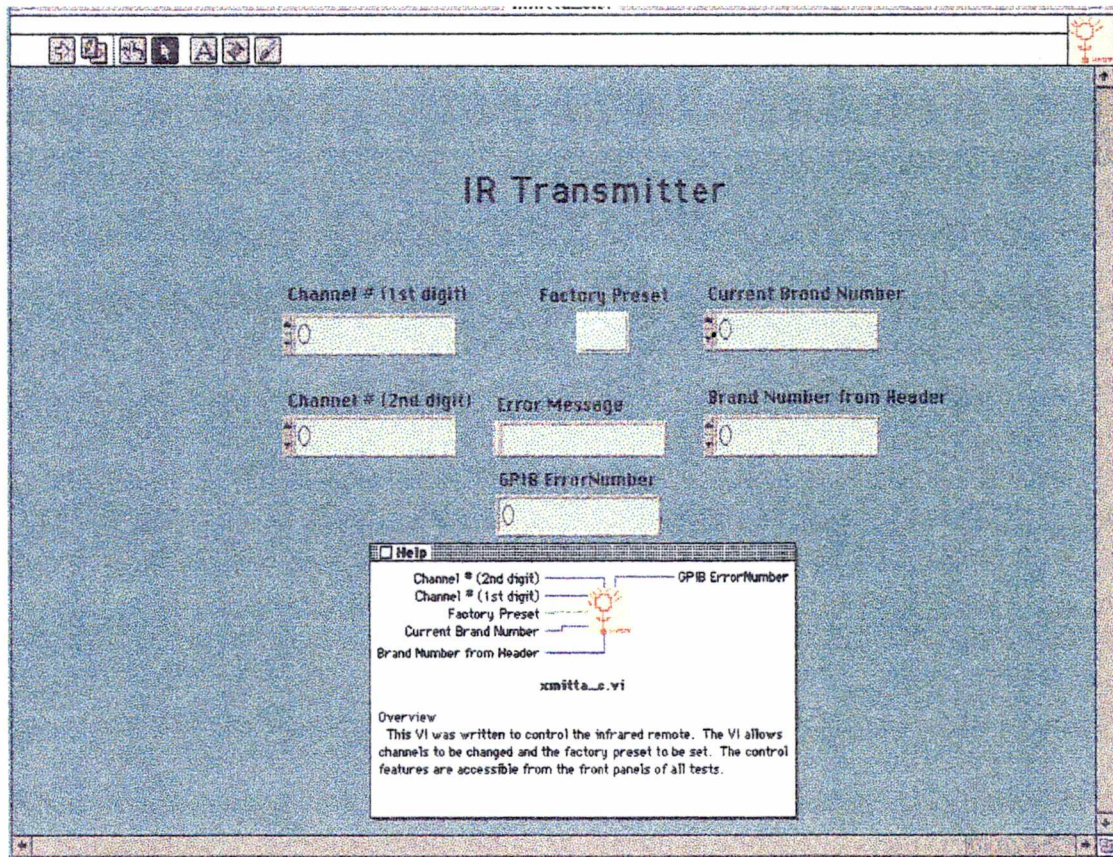


Figure 4.2 Sample Front Panel, Icon, and Connector Pane

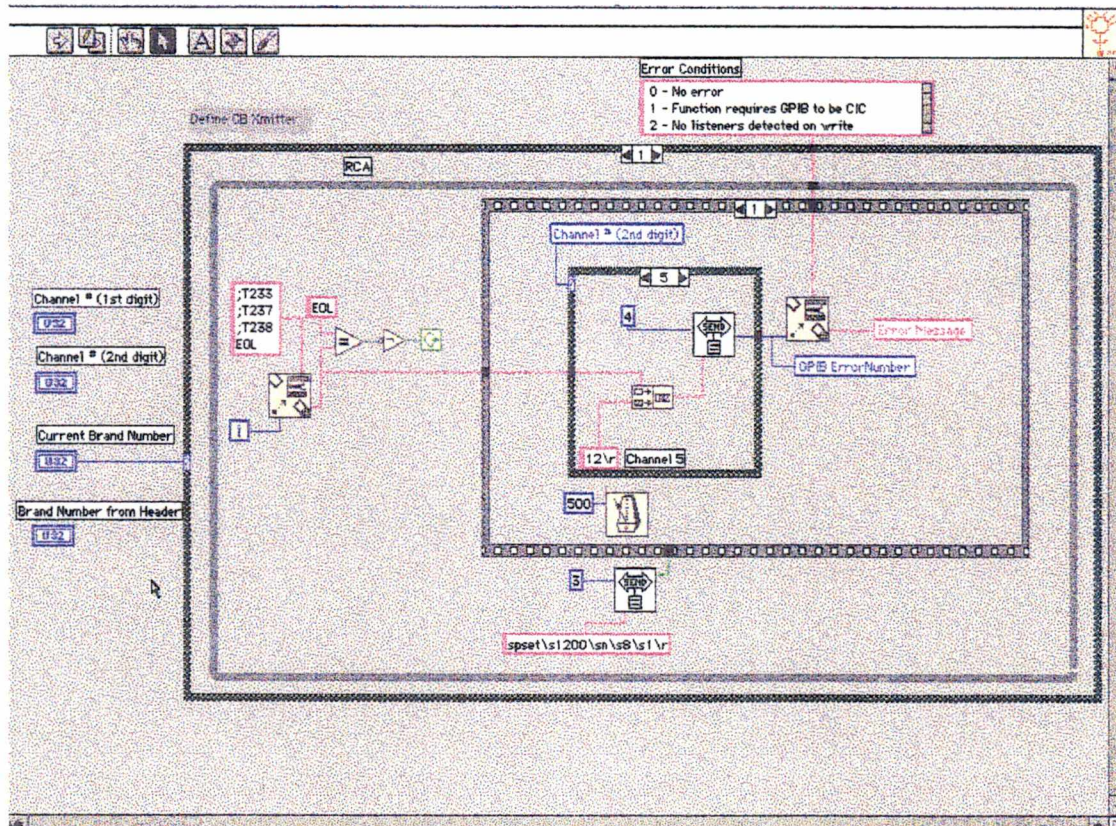


Figure 4.3 Sample of a Block Diagram

These include arithmetic, trigonometric, logarithmic, comparison, string manipulators, array tools, file I/O, signal analysis, and others. LabVIEW supports polymorphism for most of its functions. This means that functions are capable of accepting, processing, and delivering different types of data. For example, a multiplication function can accept integers, real numbers, logical data, strings, and arrays. The wiring tool automatically knows the data type of wire that is needed and displays different colors to indicate data types and different width to indicate elements and arrays of various dimensions. If a conflict exists between terminals that are wired, the wire becomes a dashed line to indicate an error [24].

LabVIEW with GPIB

Through LabVIEW, National Instruments brought GPIB to users of non Hewlett-Packard computers and devices, specializing in high speed hardware interfaces and comprehensive software. The GPIB VIs for LabVIEW adhere to the IEEE-488.1 specification. LabVIEW utilizes two GPIB libraries for communicating with instruments. These are the GPIB-488.2 VIs and the Traditional GPIB VIs.

The GPIB-488.2 VIs, when used with IEEE-488.2 compatible devices, decrease differences between instruments and improve the predictability of instrument and software behavior. The boards strictly adhere to the IEEE-488.2 standard. The GPIB-488.2 VIs are classified into five functional categories: single-device, multiple-

device, bus management, low-level, and general. The traditional GPIB VIs are compatible with IEEE-488 and IEEE-488.2 instruments [25].

LabVIEW is well suited for GPIB control of instrumentation and measurement equipment. National Instruments has supplied many LabVIEW instrument driver models, VIs, and VI templates. Writing LabVIEW instrument drivers simplifies instrument control and reduces test program development time by eliminating the need to learn the low-level programming protocol for each instrument. Furthermore, since the software's approach to programming is intuitive, it is an ideal development tool. The VIs allow fast testing and verification of remote capabilities of an instrument [26].

LabVIEW supplies recommendations for developing instrument drivers in the on-line help and reference manuals. These walk the developer through designing the instrument driver structure and VI hierarchy, organizing the VI driver, and maximizing modularity [25]. National Instrument's technical support is yet another reason to develop drivers with LabVIEW. LabVIEW has a superb technical support staff. The consultants were very enthusiastic, knowledgeable, and prompt [24]. LabVIEW also furnishes an instrument driver design example in its Instrument I/O manual to assist the novice developer [25].

The Cleanroom Development Process

Motivation

In today's highly competitive workplace, management is constantly looking for ways to improve quality and productivity; thus, reducing costs. This focus on eliminating rework and costly inefficient work processes have left no business untouched: including the area of software development [27].

Traditionally, defects are viewed as unavoidable and intricate defect removal techniques are designed into the code development process. Hence, the software goes from development to the unit testing and debugging stage, then on to function and system testing for further debugging. Programmers are usually encouraged to move code quickly into execution, so the debugging process can begin. Developers have begun to realize however, that money and time are better utilized producing zero-defect code on a first run rather than trying to clean up errors later [28].

The Cleanroom software engineering process is founded on the principle of incremental development iterations, where each successive increment includes the accumulation of all previous increments' developments. Hence, the increments accumulate through the last increment and inherently into the final design. It is desirable when possible to place items that have the highest risk in the earliest increments such that these would receive the most cycles of correctness verification and certification testing. The incremental development process lends itself well to software

development in that at onset, the customer more often times than not, does not know exactly how the system should behave so it allows the customer to interact with developers to provide input during the intermediate steps in the process. Hence, change could occur if necessary, and quality and progress can be measured throughout the process [29].

Results from organizations that have used Cleanroom are remarkable. Measuring from first execution, typical results for traditional design methods are 25 to 35 errors per thousand lines of code (KLOC); whereas for Cleanroom projects, results average 2.3 errors per KLOC. The number represents all errors found in testing, measured from first ever execution through test completion; it is the average number of residual errors remaining after correctness verification [28].

Components of Cleanroom

As mentioned earlier, with Cleanroom; correctness is built into the software through the incorporation of specification, design, and verification. The formal methods of the Cleanroom process are depicted in Figure 4.4. At the top of the diagram, sit customer requirements. This structure implies a stringent top-down design. Each component of the process in its pure form will be discussed, followed by a discussion of how these were modified to the graphical format of the ATF project.

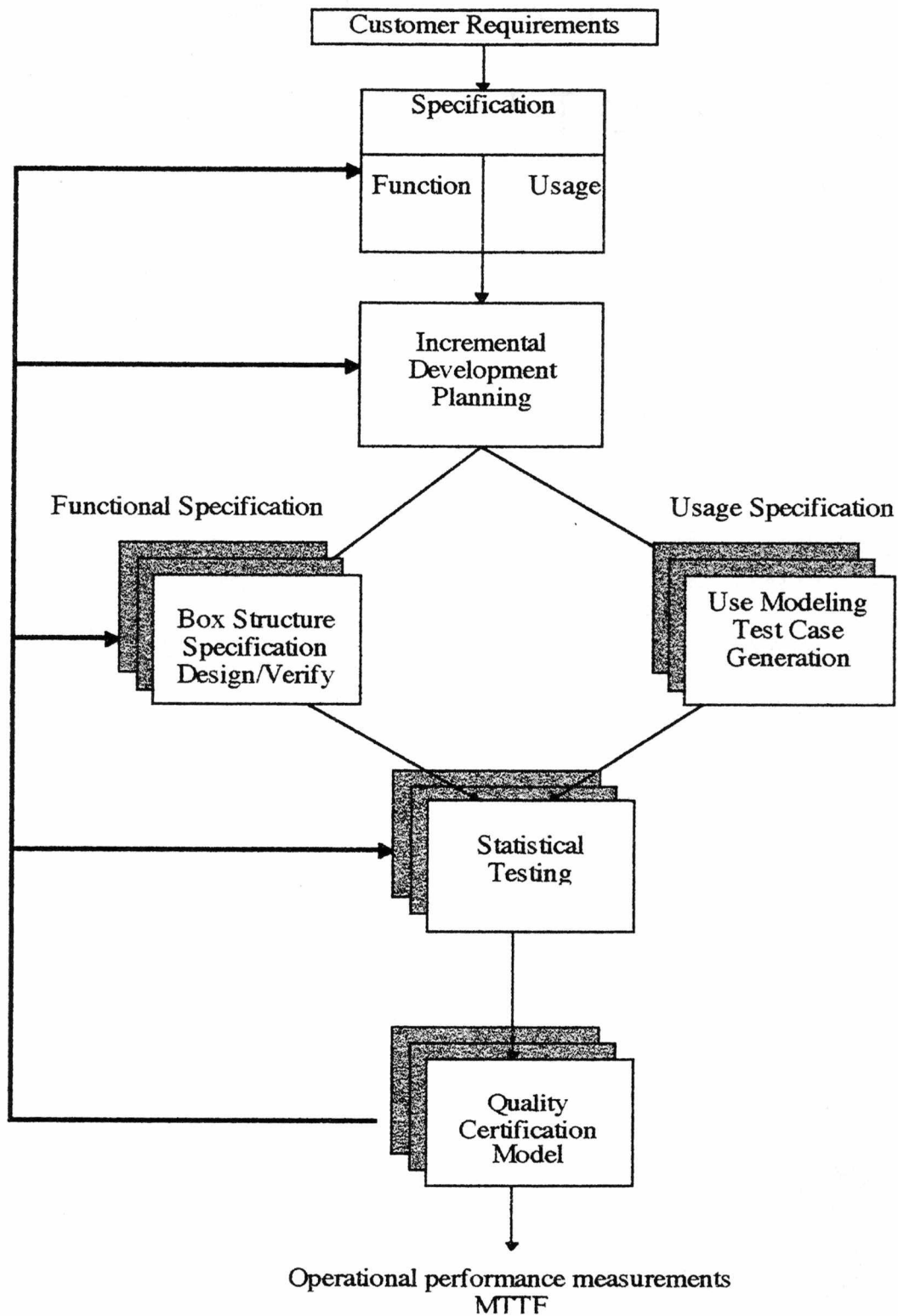


Figure 4.4 The Cleanroom Software Engineering Process

The Process

System Requirements

The purpose of the system requirements document is to define how the system is to behave. This is done at a high level so as not to include any facets of design. The document serves as a starting point as well as an agreement between the customer and the developer. This exercise helps the customer to be thorough in defining how the desired system should behave. The customer will have the opportunity at each increment's end to provide feedback on the system through that point. This may lead to modification of the system requirements.

Specification

Specification is the top-level of the Cleanroom process. The functional specification and the usage specification are defined at this point. Although, the development team and the certification team usually produce these, it is not uncommon for there to be a stand-alone specification team for large projects [28].

Functional specifications define the necessary external system behavior in all circumstances of use. It is the basis for incremental software development. The use of box structures is highly recommended for this step.

Box structures allow objects to be defined and refined, resulting in a usage hierarchy of objects that can be used and reused throughout the system. Ultimately, system behavior is defined through these boxes. Initial box structures (black box)

often reveal misunderstandings of customer requirements. With this tool the gap is found and filled at an early stage in the development rather than at later, costlier stages. The boxes also possess referential transparency meaning that the information contained within a box is enough to define its refinement without having to use any other box. [28]. The box structure process is a systemic way to develop object-based systems. It begins with the black box that specifies the necessary external system behavior in all circumstances of use. These are defined in terms of stimuli, responses and transition rules that map stimulus histories to responses. The state box proceeds from the black box and is refined from and verified against it. It also contains the state data to satisfy black box behavior. The clear box is refined from and verified against the state box. New black boxes often stem from this procedure [30].

Usage specifications define usage situations and their probabilities for all possible system usage. This is the basis for generating test cases for incremental statistical testing and quality certification. Usage specification involves sampling the population of all possible executions by users according to the frequency with which the executions are expected to occur [28].

Incremental Development

Increment planning begins with defining an initial plan for developing increments that accumulate into the final system upon completion of the last increment. Size and complexity of an

increment's task drive the time allotted for the design and verification for that increment. As seen in Figure 4.5, incremental development is a life-cycle process model. This allows customers to provide feedback on the evolving system by operation of user-executable increments. Each increment encompasses specification, statistical usage testing, verification, certification and implementation of customer feedback. It is not uncommon for early increments to be enlisted as system prototypes, whereas they are used to validate system requirements and functionality through customer feedback. It follows of course that changes may occur. Naturally, these may cause revisions in schedule, priorities, and functionality; however, the changes tend to be orderly and methodical [30].

Correctness Verification

The Cleanroom process replaces individual unit testing and debugging with team correctness verification. The tool reveals its advantages through the human psychology factor; particularly, that programmers are inspired to deliver error free code for first execution when confronted with being evaluated by peers [30]. Besides self pressure, the process calls for verification of the accuracy of every control structure by deriving the function it computes and records it to the intended value as recorded in the design. There are basically three control structures. These are the sequence, alternation, and iteration structures. One condition exactly must be checked to verify a sequence. Alternation correction verification need exactly two conditions checked. An

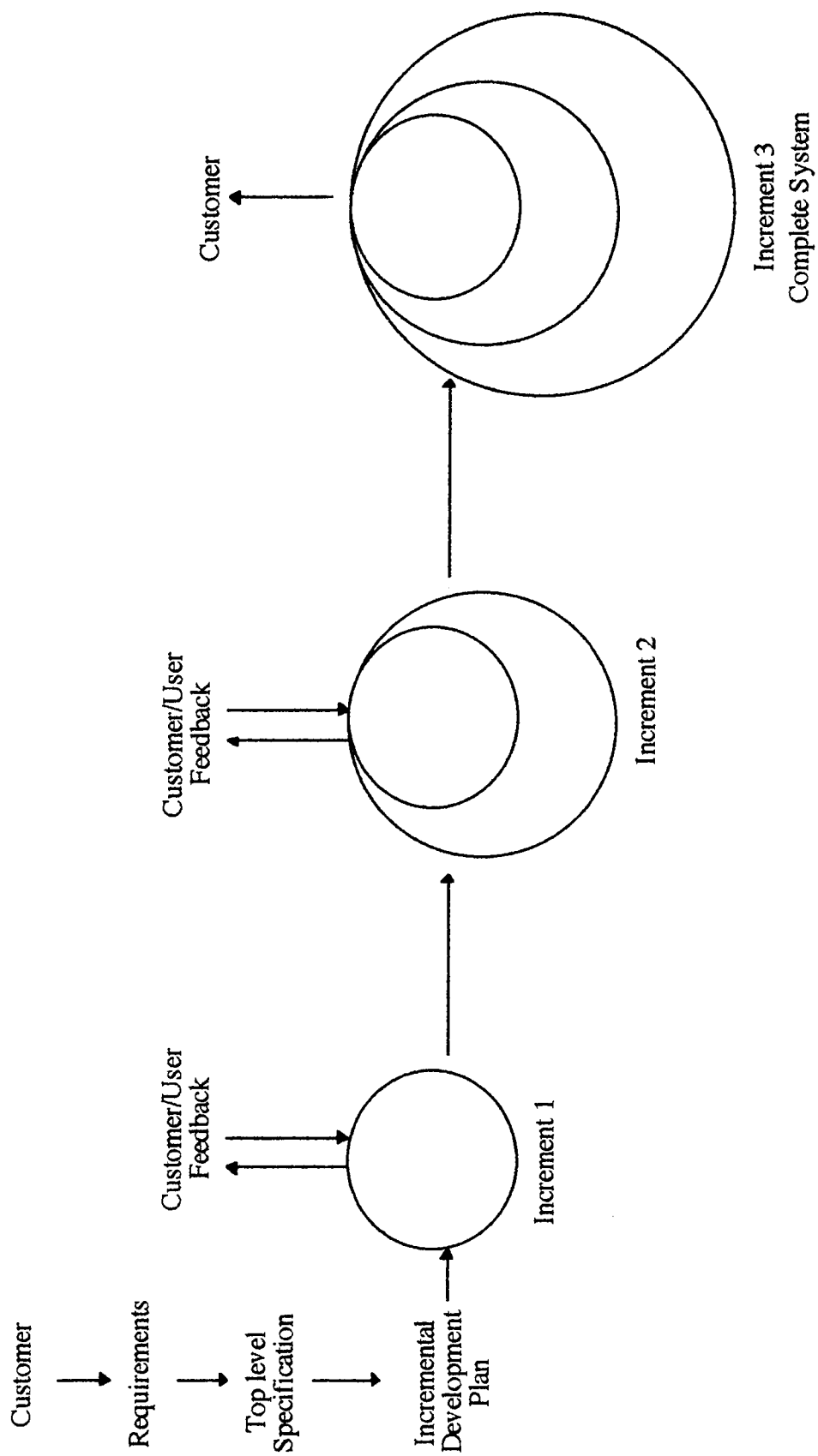


Figure 4.5 Incremental Development Model of Prominent Life Cycle

iteration requires case analysis and the verification of exactly three conditions. Hence, verification becomes a process and a verification team is able to verify every line and design of code. Since every correctness condition is verified with this process, near zero defect code results later at runtime [28].

Compared to unit testing which only check the effects of executing selected test paths out of numerous possible paths, the Cleanroom approach can verify every possible effect of all data. This stems directly from the fact that Cleanroom verification is based on function theory [28].

Quality Certification

Cleanroom quality certification has three steps, which advance in correlation with code development. In the Cleanroom process, certification is based on predicted usage by customers. Usage probability distributions are developed to define system inputs for all aspects of usage. Test cases are then generated based on random sampling of usage distributions. The number of test cases required for statistical soundness is dependent on the batch size and desired inspection level; MIL-standard 105D provides reference tables for proper determination of these [31]. Every test represents actual usage that simulates user experience. Verified code is released to the certification team for first-run execution. Each test case is executed and assessed by comparing correct results (system specifications) to actual outputs using common inputs. The execution time to failure is recorded for each failure. The times

represent the quality of the sample of possible user executions. The system's quality is then computed according to a quality certification model. The mean-time-to-failure is also determined. Therefore, software quality becomes measurable and its former empirical qualitative measure of excellent, good, fair, or poor is eliminated [28].

ATF Implementation of Cleanroom

Motivation

In a paper by Hausler, Linger, and Trammell [30], the partial implementation of the Cleanroom process as opposed to its complete realization was discussed. It was reported that substantial gains in quality and productivity had occurred in these partial Cleanroom implementations. There were three implementation phases mentioned. These were introductory, full, and advanced phases. It concluded that benefits could be reaped from such systems and that an implementation of Cleanroom could combine elements from various phases as necessary and appropriate for the project [30].

Such was the case in the ATF project. The team agreed that a methodical approach to the development of the code was needed and wanted. As fortune would have it, the University of Tennessee's Computer Science department was home to a leading authority on the Cleanroom process. Dr. Trammell lead the team through training sessions on Cleanroom and became a resource to the team. The team struggled with how to fit the project into Cleanroom as it existed in a timely manner. The greatest challenges

arose from the application of Cleanroom to a graphical programming environment like LabVIEW. There was no documented benchmark for the implementation of the Cleanroom process with a graphical language, let alone LabVIEW. It was clear that the team was navigating uncharted territory. Coupled with the time constraints of the customer's project deadline, the team moved to retain elements of Cleanroom that seemed to fit the project and offered the greatest benefit. So some of Cleanroom's elements were omitted or altered to fit the project. A discussion of the adaptations, omissions, conventions, and work norms that were used with the resulting partial Cleanroom process will be discussed. The documents generated for the project will be considered also.

System Requirements

The ATF system requirements document thoroughly defined the needs of the customer within the confines of the contractual agreement. It began with a high level description of the purpose of the project in the introduction. Next, without going into the design, the user interface for the system was defined. Described here are the modes of operation, interactive hierarchical menu structure, and measurement criteria. The test environment was defined next by describing signal sources, equipment behavior, system interface, and the programming platform. Next the test methods to be employed were defined. The procedure for retention of measurement data was explicitly described next. It defined how data were to be stored separately for the two areas of interest, the file naming convention, and the format of the file records. Finally, it

described the ATF documentation that is to be delivered with the software at the completion of the final increment. These are the System Requirements Specification, User Interface Description, Cleanroom Process Description, Incremental Development Plan, and Software Functional Description [6].

Specification

The ATF team utilized the box structure and Box Definition Language to incorporate functional specification into the project [27]. Figure 4.6 shows a LabVIEW black box that was used in the project. On both the front panel and the block diagram, controls or stimuli are placed on the left side and the indicators or responses are on the right. These are separated only by the BDL that describes in a pseudo textual sense, the behavior of the system without incorporating state data. The ATF state box was built from the black box. It added major structures such as loops, cases, shift registers, and some functions. The BDL was also modified to include state data. Figure 4.7 shows an example of a state box.. Finally, the clear box evolved from the state box. The clear box as shown in Figure 4.8, has full functionality and undergoes correctness verification and certification testing.

The ATF team elected not to employ usage specification. Rather, a thorough and rigorous user interface specification was developed. This included top-down storyboard walkthroughs to ensure appropriate functionality in system usage. The user interface specification will be discussed later in this chapter.

Security Panel

User Name

Access Granted

☐

Password

Experienced

☐

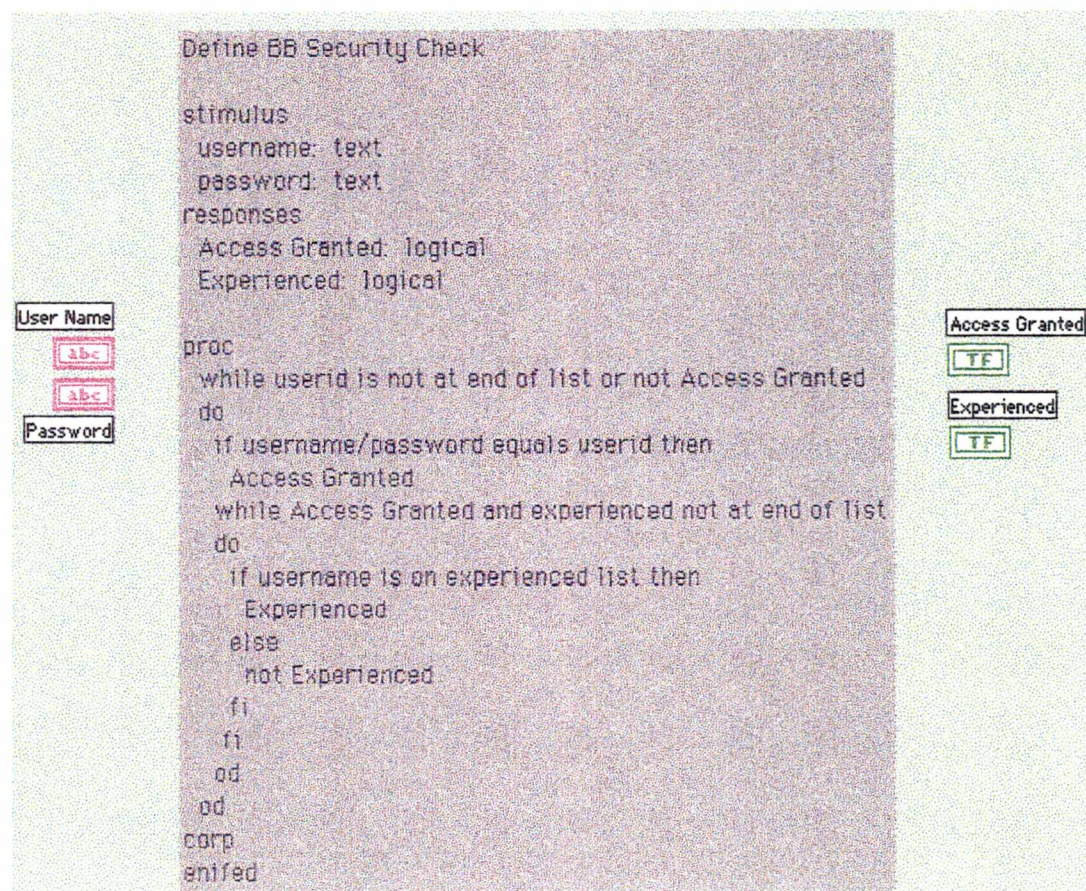


Figure 4.6 Front Panel and Block Diagram of Black Box

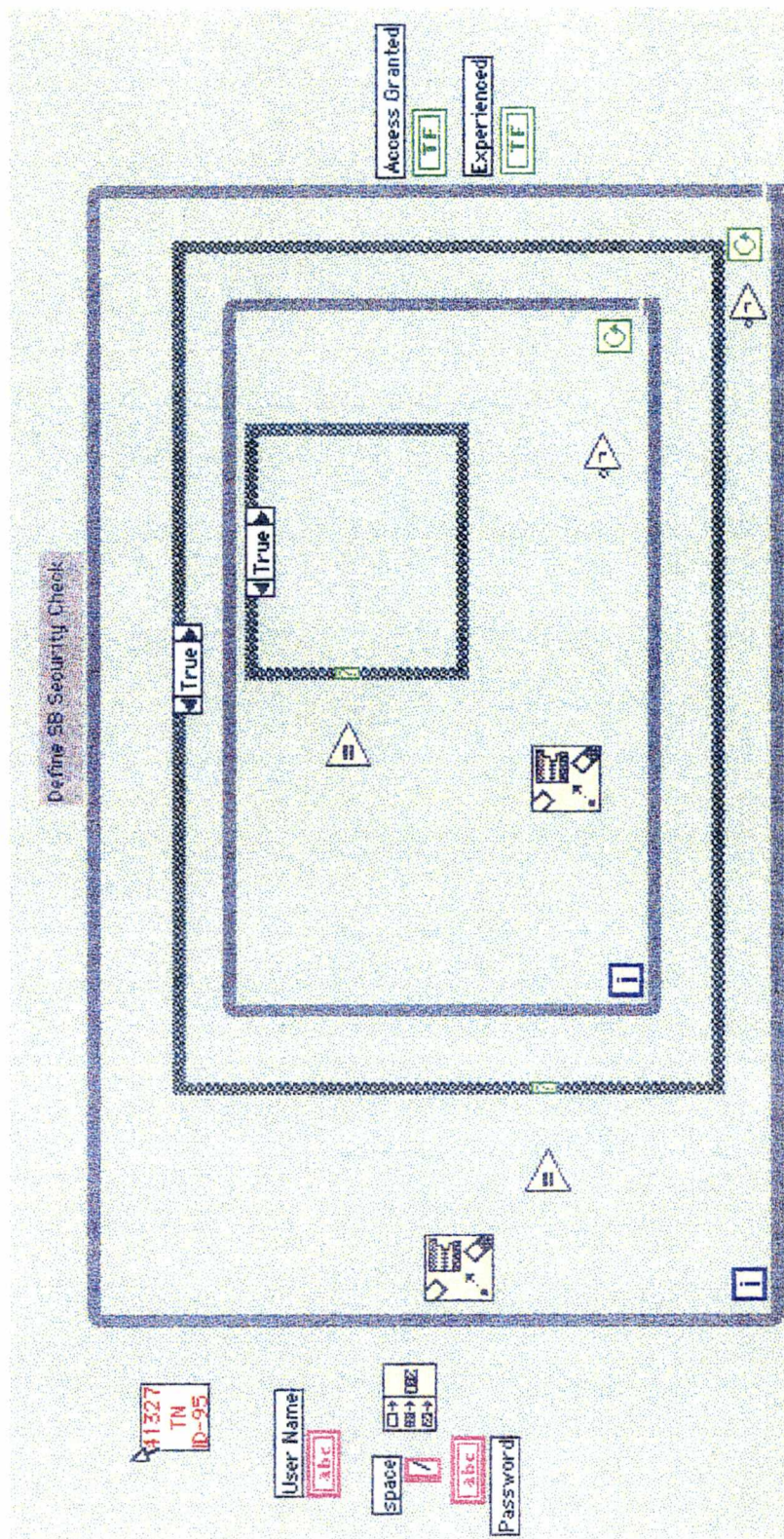


Figure 4.7 Block Diagram of State Box

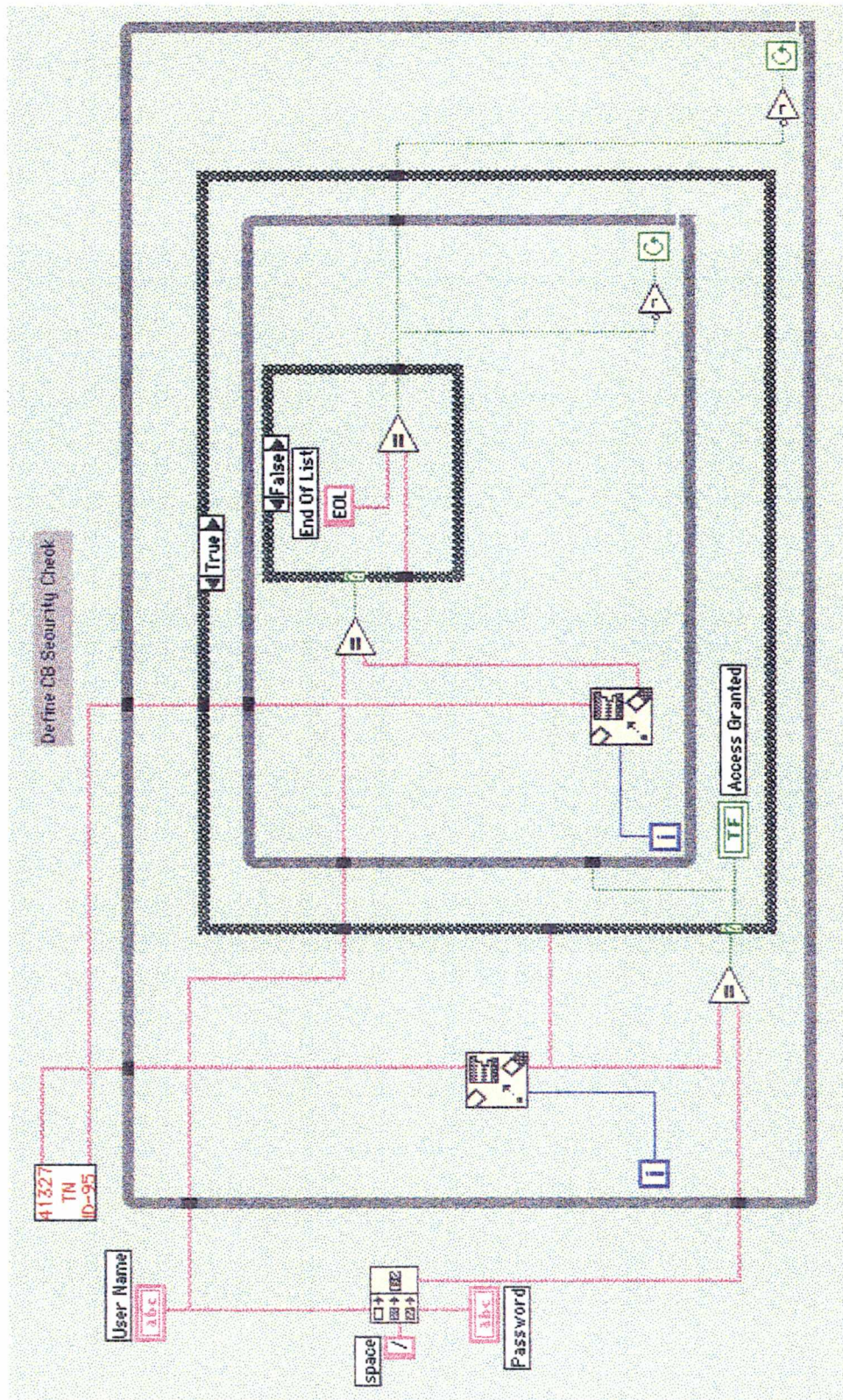


Figure 4.8 Block Diagram of Clear Box

Design by Incremental Development

The team developed an initial incremental development plan that spanned six increments. Subsequent schedule revisions followed till it was decided that nine increments was the proper length for the project. Factors that affected the number of increments and the lengths of each were personnel level changes, incorporation of a BIT system, and shifts in customer priorities. An early incremental development plan is shown in Figure 4.9. Assignments for each increment were issued. Developers then set about to write the code. This began with individuals describing the black box behavior of a subsystem. This may have included co-development with designers of interfacing black boxes to determine the characteristics of common stimuli and responses for passage of data through connections at the connector panes located behind the descriptive icon. Each control and indicator had thorough descriptions that could be accessed through LabVIEW's help tool. Further documentation to assist the user was located in LabVIEW's Get Info. The Get Info contained information describing the VIs eventual future behavior, input requirements and modification history. In Figure 4.10 the information contained in the Get Info panel is shown. All documentation was furnished by convention to ensure value added software for the customer. As mentioned before, this step included the use of box definition language which was devoid of any history or state information. As an extension to the black box, this aforementioned information was however, included in the state box. LabVIEW shift registers, control

INCREMENTAL DEVELOPMENT PLAN

Increment #	Function	Start Date	End Date
1	User Interface to Test Level Cleanroom Software Engineering Training System Requirements Document Story Board (User Interface) Incremental Development Plan Cleanroom Process Description UTMacintosh LabVIEW 3.0 Setup	5/01/94	7/15/94
2	RF Sensitivity Front Panel Set User Interface Conventions UTSun Workstation Setup Philips Sun Workstation Setup	7/18/94	8/10/94
3	RF Sensitivity Test TEK TSI 8150 Driver TEK 2714 Spectrum Analyzer Driver Cross Point Router Driver X-Terminal Setup Network Printer Setup System Loss Level Table	8/15/94	11/30/94
4	Minimum Line Voltage Test Cross Modulation Test Horizontal & Vertical Resolution Test Noise Rejection Test Philips PM5640 Driver RE540 Driver Magtrol AC Voltmeter Driver Staco Power Control System Driver Horizontal & Vertical Resolution Using Direct Video Input Test	12/05/94	1/18/95
5	Dipole Antenna Operation Test	1/23/95	3/01/95

Figure 4.9 ATF Incremental Development Plan

Overview:

Security Check VI accepts the strings username and password from the Login Menu. The VI checks the username and password to determine if the user is authorized for access and if the user is experienced. The VI is case and space sensitive. Regarding letters, the VI expects lower case.

inputs:

User Name: text; Password: test

outputs: Access Granted: logical; experienced: logical

Since this is a SubVI of a larger algorithm, the user needs only to follow the instructions of the VI (Login Menu) that called Security Check.

Author:

Eric Wrushen, ATF Team originator 7/11/94

Modification History:

7/11/94 VI created

7/14/94 Modified per Verification Feedback

2/15/95 Changed string constant for users & passwords to sub-vi

3/17/95 Changed users & passwords from sub-vi to Unix text file

Figure 4.10 ATF LabVIEW Get Info Description

structures and icons were added to the stimuli and responses in the state box design. This format represented the high level translation of design from BDL to LabVIEW. The part of the BDL that corresponded to a particular control structure was inserted in the structure's description box. Thus, the BDL described the behavior of the structure's future internal logic. This helped modularize the BDL which made verification of logic less complicated. By convention, the state box contained no wiring. The granular LabVIEW design implementation was reserved for the clear box. The clear box which was the state box refinement, was the complete implementation of the design in LabVIEW that would undergo the rigors of correctness verification, certification testing, and be integrated into the system for delivery.

Verification

Since LabVIEW's graphical structures mirrored those mentioned in the earlier discussion of correctness verification, the box structure process was adopted. It was also however, augmented such that the BDL could be used to cross check the LabVIEW implementation. In addition to checking for a match between code and BDL, the accuracy of each was checked. The BDL used with LabVIEW proved to be well suited for the application of Mill's method of box reduction where a clear box is reduced to a state box and it to a black box. Each reduction should not affect the stimuli or responses [31]. This method was employed in verification; hence, each structure had to be checked. Verification

proceeded on a per VI basis. Completion of verification for a particular VI was signified by the signature of the verifier, an ATF team member other than the VI designer. In the event of exceptions, the designer was required to correct them and turn in the verification sheets for inclusion in the process description document. This was done largely on an honor system, where individuals were entrusted to do the right thing. On the other hand, severe exceptions, usually logic errors or flagrant omission of conformity to conventions, led to a return of a verification sheet without a signature. Hence, the VI would have to be corrected and re-submitted for verification. A standard verification sheet was constructed to ensure uniformity in the process. It is shown in Figure 4.11. As mentioned previously, in addition to checking for code correctness, convention conformity was checked. This included the checking of required documentation, cleanliness in wiring, proper font sizes and colors. LabVIEW's flexibility permitted the team to use colors to distinguish between the three box structures, in addition to the naming convention. Pastel yellow was used for black boxes (BB), olive green for state boxes (SB), and turquoise blue for the clear boxes (CB). After ensuring that VIs were correct and conformed to the established conventions, the VIs were ready for first-time execution; at which time they would be certified.

Designer's Name _____ VI Name: _____

No. Item	Comments
1. COLORS	
a. BB: Pastel Yellow (R=255,G=255,B=204)	
b. SB: Olive Green (R=205,G=255,B=102)	
c. CB, etc.: Grey (R=127,G=191,B=191)	
d. i. ALL Block Diagrams=GREY (229,229,229)	
ii. ALL TEXT=BLACK	
iii. BDL= GREY (204,204,204)	
2. SPACE (BB,SB,CB)	
a. Vertical Alignment	
b. Horizontal Alignment	
c. Plenty of GREY space on Block Diagram	
3. DESIGN (CB)	
a. BDL and Code match (precision)	
b. BDL accomplishes task (accuracy)	
c. BDL box shrunk to display only: "define..."	
d. GET INFO: Overview, Instructions (if necessary), I/O (if necessary), Authors' names, Modification History	
e. DESCRIPTION BLOCKS: use BDL with extra text if necessary	
f. HELP Boxes should be descriptive; NO BDL	
g. NO type conflicts; i.e., no grey dots	
h. State Variables & Shift Register Variables appear in BDL	
4. MISCELLANEOUS	
a. Wiring cleanliness	
b. Descriptive Icons	

ADDITIONAL COMMENTS

Verifier (print) _____

Verifier (sign) _____

70

ATF Certification

For text languages, errors per KLOC is a popular unit of measure. With LabVIEW a benchmark unit of measure that is translatable by some proportion to KLOC, and makes sense does not exist. Secondly, having chosen not to apply usage specification, usage probabilities could not be calculated; hence, random sampling of test cases based on usage could not be done.

The ATF team's certification was thorough run-time testing that included the activation of all stimuli to check for accuracy in responses. All of the increment's VIs were integrated into the existing system. By convention, during certification, defects were noted and fixed off-line, provided the exception did not cause testing to cease. In such an event, the affected VI would be removed or certification would be stopped until the problem had been corrected.

Delivery

Delivery of increments to the customer involved all team members. Deliveries consisted of demonstrations of test systems and instrument drivers. Each ATF team member presented code that he or she developed. The customer had the opportunity to ask about functionality and aesthetics. As expected, this process usually generated requests for changes in software functionality which invariably meant design changes. On occasion, deliveries disclosed limiting capabilities of hardware which prompted reconfiguration of the affected part of the system or the redefining of test systems.

The feasibility of the feedback was considered and was either implemented, postponed to a later increment, or disregarded in accordance with the customer's wishes. System alterations were presented during the next increment to the project leader or as part of the next increment.

Postmortems

In a continual team-improvement effort, postmortems were held following each increment. These meetings allowed the team to discuss the things that were done correctly and the improvement opportunities for succeeding increments. Quite often it was during postmortems that conventions were appended, work processes altered, or systems re-analyzed to ensure that current best approaches were being used. Always an issue of concern was the customer's impression of the presented work.

Documents

Several documents were generated from the project. These are the System Requirements Document, User Interface Specification, Cleanroom Process Description, and the Incremental Development Plan. The first and last documents listed have already been described; therefore, the ensuing discussion will be limited to the purpose and content of the User Interface Description and the Cleanroom Process Description.

User Interface Specification

The purpose of the User Interface Specification is to document the physical appearance of the components of the system, the

motivation for such appearance, and all system stimuli and responses that are due to user interaction with the component. It also tells how the stimuli and responses are generated by the user's action [32].

The ATF User Interface Specification included conventions for the test panels. These conventions describe all interfaces that are visible to the user. The system has a definite top-down design to its structure. The philosophy behind the hierarchy is addressed here. The philosophies behind including various components and options in tests are listed. These include error handling, instrument driver conventions, and conventions for logging data. Finally, conventions for on-line help and other dialogue boxes are discussed.

Cleanroom Process Description

This document describes how the Cleanroom process was applied to the ATF project. This document acts as a system reference library in that it houses the information of work processes, conventions, describes how each sub document was developed. The Cleanroom process description (CPD) is organized by topic. These are Cleanroom process description, system requirements description, user interface specification, incremental development plan, top level specification, VI functional specification, correctness verification, certification process, process review and improvement, and delivery. Each topic is divided into subtopics relevant to the topic. Some of these are the purpose, process, organization, conventions, and references to prior work.

The purpose explains why the topic is important to the ATF team. The process describes what was done and how it was done for a specific topic. It lists assumptions, conventions, and policies that apply to the respective topic. The organization lists whether the entire team, sub committee, individual, or the customer participated in the development of various aspects of the topic. The reference section listed books, papers, or other materials and persons that were used as references. This document is also evolving.

Summary

The Cleanroom process undoubtedly offers a methodical approach to developing software. The application of Statistical Quality Control (SQC) and other engineering principles help to ensure this.

The absence of a method for measuring errors for graphical languages currently presents a problem for fully implementing the Cleanroom process. Clearly, if viable measurements can not be made in a process, the process can not be statistically controlled. Several possibilities were considered by the ATF team. The issue of translating the results to KLOC presented a major problem in that there was no known factor for doing such. Consequently, while the goal was to produce zero defect code, when errors were found, how were they to be interpreted? What constituted a level of excellence, good, fair, or poor? How would these levels compare to the levels for software designed with a textual language? The preceding questions will arise again and again as expectations for software

development continue to rise, the Cleanroom process and its results become more familiar, and the preference for graphical interfaces for programming continues to grow.

The next chapter will present some of the work that was produced by the author while a member of the ATF team. The discussion will consist of two instrument drivers and one test.

Chapter 5

ATF SOFTWARE

This chapter provides an in-depth narrative of some of the work that the author produced or co-produced with other team members. The discussion will begin with the instrument drivers that were written by this author. These were the RE540 and the SMT02 stereo generators. The chapter will continue with a discussion of the automated RF Overload test.

RE540

As mentioned in Chapter3, the purpose of the RE540 in the ATF setup is to test audio circuits. It allows manual operation through its front panel and remote operation via an IEEE-488 GPIB. The instrument was used for manual testing; thus, it was quite familiar to the test engineer. A driver for the RE540 existed [6]; hence, it was used as a reference for the development of the driver that was to be used in the ATF. The RE540 signal generator is shown in Figure 5.1.

There are three VI's that are used to control the instrument. In harmony with driver conventions, these are for initialization, presetting conditions, error handling and reporting, and resetting for termination. These are `reinitc.vi`, `re540c.vi`, and `r540errc.vi`.

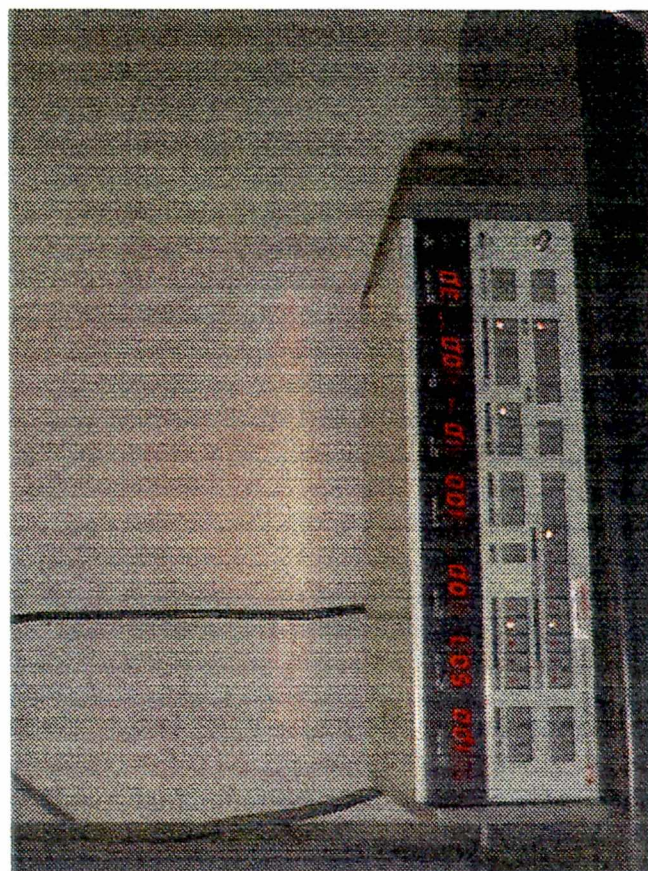


Figure 5.1 RE540 Stereo Signal Generator

Reinitc.vi

Initialization and resetting were handled by the `reinitc` VI. It was used to set the instrument to a known state. This assured the user that following operations would behave consistently. In Figure 5.2, the VI's front panel and block diagram are shown. The VI has no stimuli (external inputs) and has a single output. The *GPIB Event Code* is the error-reporting indicator that is passed from the error checking sub-VI, `r540errc.vi`. The internal operations of this VI will be deferred until its discussion. The initialization VI utilizes LabVIEW's built in *GPIB Write* VI to send the instruction set to the RE540. The *GPIB Write* VI accepts at its input ports the: address string (6 for the RE540) as denoted by the pink wire and pink string constant box; data string that contains the device's instruction set and mode of termination for the *GPIB Write* which is marked as an integer 1 for this device. The constant 1 calls for the *GPIB Write* VI to append a carriage return to the string and to send an EOI with it. The data that are sent to the RE540 instruct it to generate a 1 kHz monotone signal (R=L) with a 25 kHz deviation (100% modulation) without DBX encoding and with the SAP (Second Audio Program) off. The *GPIB Write* VI returns a 16-bit status boolean array where each bit defines a state of the GPIB Controller. Bit 15 is set when an error occurs. The status is sent to the error checker, `r540errc.vi`.

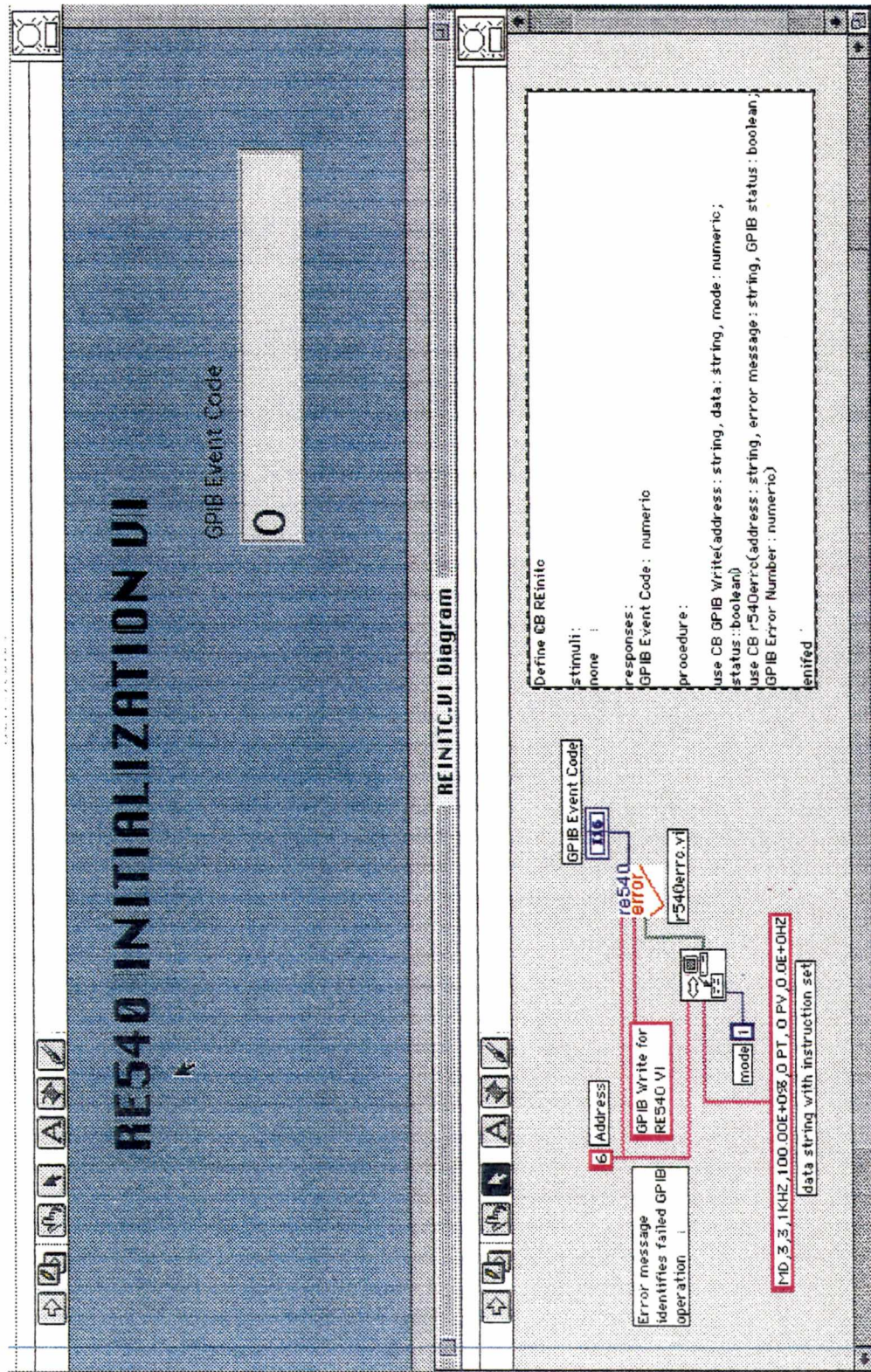


Figure 5.2 Front Panel and Block Diagram for Reinitc VI

R540errc.vi

This error checking VI was constructed from a prototype that was initially designed for the TSI-8150. The original VI was co-designed with another ATF team member [33]. The goal was to make a generic error checking VI that could be easily customized for an instrument by simply changing the name of the instrument in the error message header; thus, increasing productivity through code reuse.

In Figure 5.3, the VI's front panel is shown. The VI has 3 inputs for stimuli and has a single output. The *Address String* control accepts the address of the device that was most recently invoked via the GPIB. The *Error message* accepts the string information that will be used as the error message header. The last stimulus is the *GPIB Status* that accepts a 16-bit status boolean array where each bit defines a state of the GPIB Controller. Bit 15 is set when an error occurs. The *GPIB Error Number* is the error reporting indicator that is passed to higher level VI's.

The block diagram for this VI is depicted in Figure 5.4. The address is sent to LabVIEW's *GPIB Status* VI which reports the status of the GPIB controller at the input address after the previous GPIB operation. It reports back the GPIB error number that is wired to two separate *Pick Line & Append (PL&A)* functions as indexes. The *PL&A* functions choose the string on the line that corresponds to the index from a multi-line string input. It appends the empty string for each *PL&A*. The upper *PL&A* outputs the error

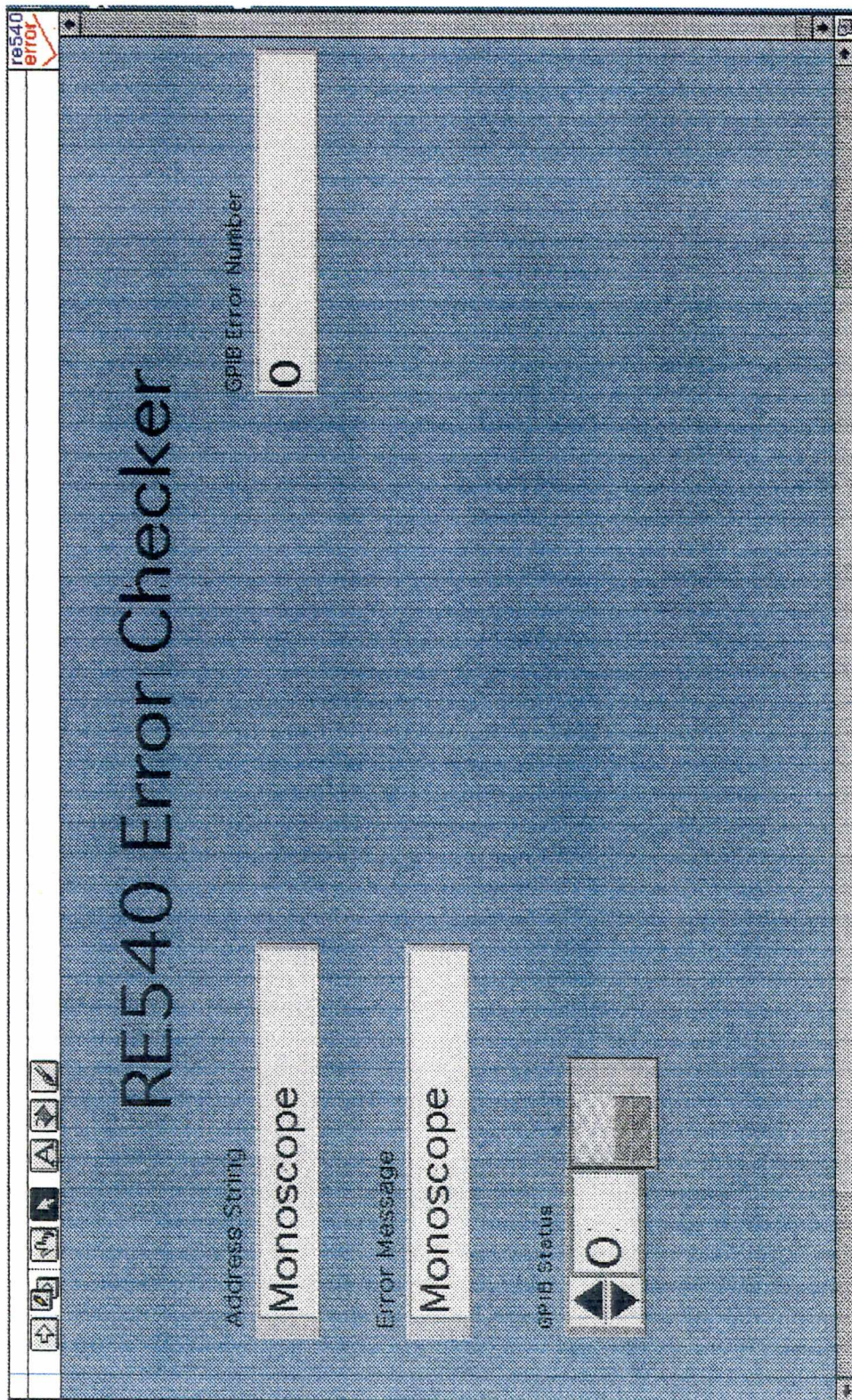


Figure 5.3 Front Panel of R540errc VI

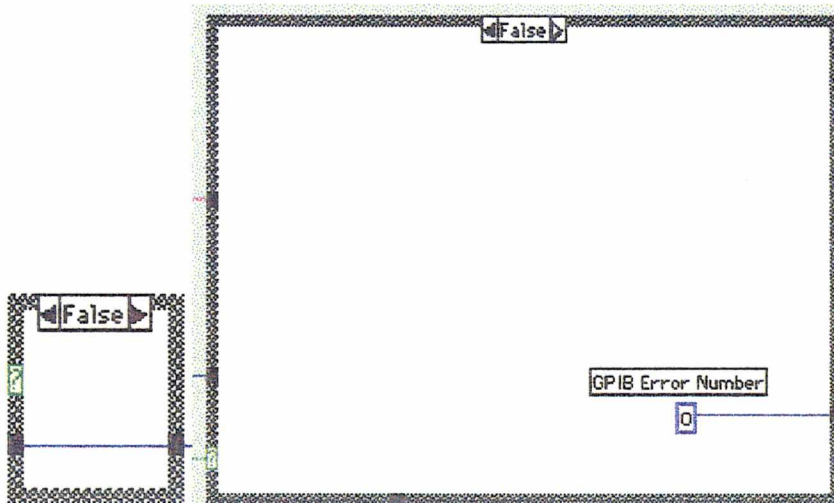
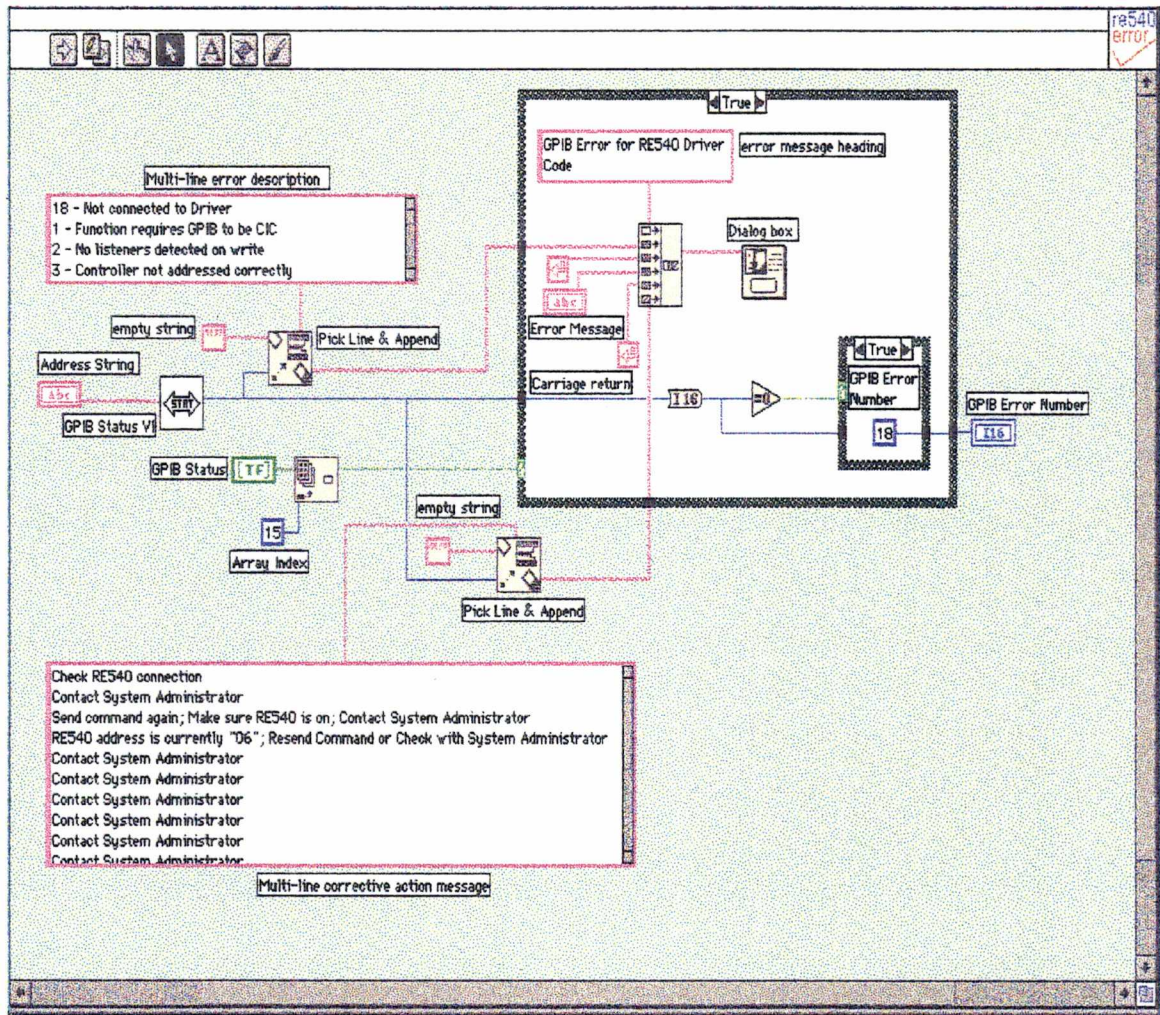


Figure 5.4 Block Diagram for R540errc VI

description associated with its respective *GPIB error number*. The lower *PL&A* outputs the appropriate corrective action that corresponds to the indicated error. The fifteenth bit of the 16-bit boolean array *GPIB Status* is checked by first isolating with the *Index Array* function and then wiring bit 15 to the selector located on a logical case structure. If the bit is true, an error was found and the logic of the true case is executed. The logic for this case appends to the error message heading the error number, error description, the error message which identifies the GPIB operation that was attempted, and the suggested corrective actions to take. The concatenated string is sent to a dialog box which will halt execution and appear for the operator to see. The GPIB error number is checked for a 0 so that an 18 can be passed as the error number. Otherwise, the *GPIB Error Number* is sent as it is passed from the *GPIB Status VI*. This was necessary because by convention, 0 was reserved to indicate that no error was found. The false case for the large case structure sends 0 for its *GPIB Error Number* to indicate that no error occurred.

Re540c.vi

The task of presetting and audio control during a test was handled by the main VI, *re540c.vi*. As shown in the front panel of Figure 5.5, the VI is required to support the control of many functions, thirteen to be exact. The VI's front panel was designed to look like the RE540 interface panel: this included adopting the names for the controls. Seven of these are stimuli for defining stereo operations. Five controls are stimuli for defining the SAP

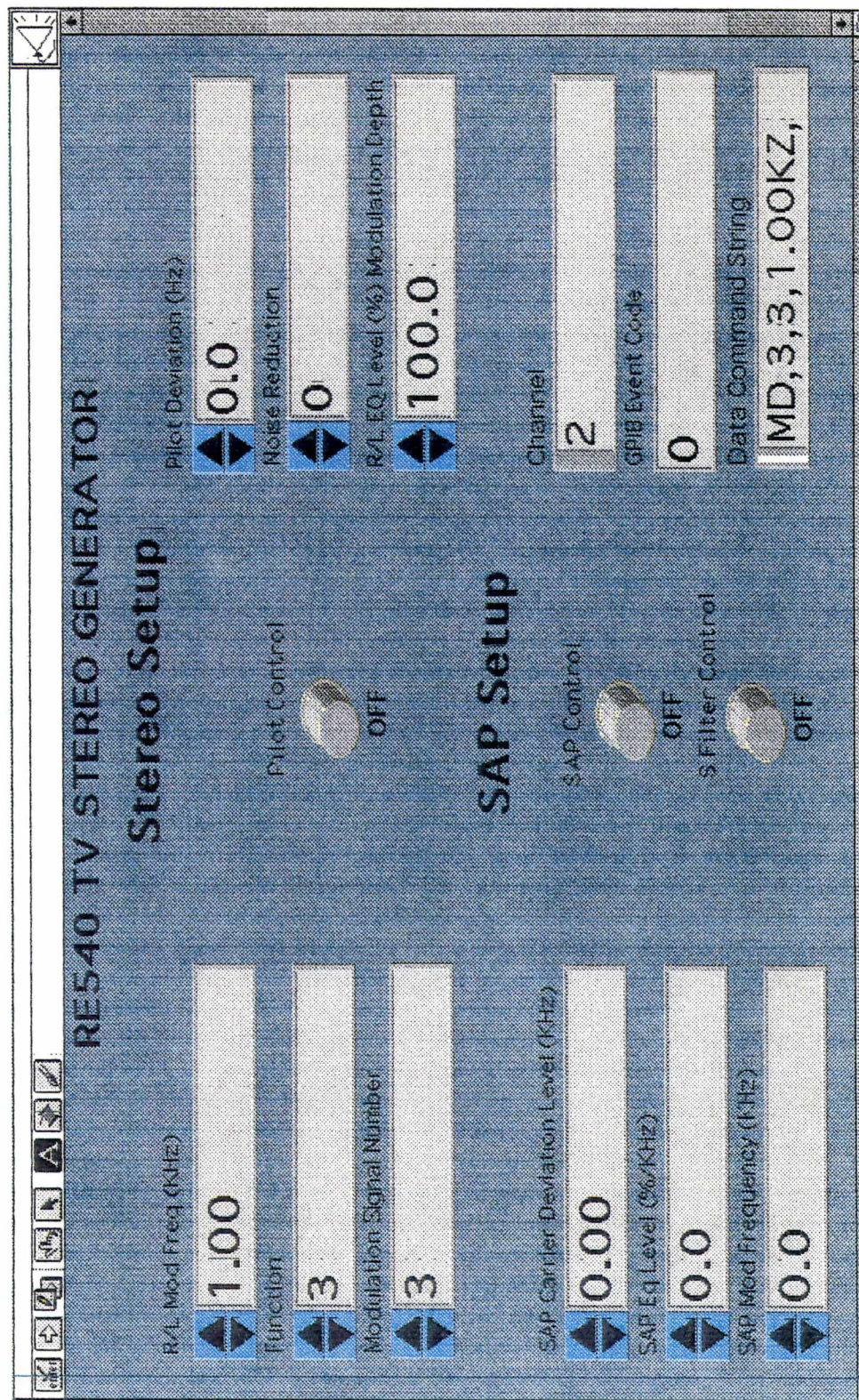


Figure 5.5 Front Panel of Re540c VI

setup. The *Channel* control allows use of the RE540's internal VHF modulators for channels 2, 3, 12, or 13. The user enters one of the aforementioned channels to use this function. The responses are represented by two indicators. These were the *GPIB Event Code* and the Data Command String.

The modulation source command (MD) is used to define a single tone signal. To fully describe the signal five parameters must be defined. The *Modulation Signal Number* accepts as valid input integers 1-4. This parameter allows for the selection of modulation sources. The source choices are:

- 1-Left (L) stereo only
- 2-Right(R) stereo only
- 3-L=R Monotone
- 4-L=-R

The *Function* control accepts as a valid range integer inputs 0-4. The *Function* control allows selection of the desired stereo mode according to the following list:

- 0-Modulation signal off
- 1-Left (L) stereo only
- 2-Right(R) stereo only
- 3-L=R Monotone
- 4-L=-R

Control of signal frequency is passed through the *R/L Mod Freq* control. This is a numeric control with a valid range of (0.05-15.0) kHz for its frequency input. These steps must be in 0.05 kHz steps otherwise truncation to the nearest lower 0.05 kHz resolution is coerced by the instrument. The *R/L EQ Level Modulation Depth* numeric control accepts inputs 0 %-99.9 % in steps of 0.1 % and values 100 %-300 % in steps of 1.0 %. The RE540 computes the equivalent modulation depth. For example an input of 200 % will cause a 25 kHz and 50 kHz deviation in the mono channel and stereo channels respectively. The *Noise Reduction* numeric control accepts valid inputs 0, 1, 2, or 3. This enables the operator to select the required noise reduction mode:

0-Noise Reduction Off

1-DBX encoding

2-75 μ sec pre-emphasis

3-DBX- 75 μ sec pre-emphasis

The *Pilot* control is a boolean that accepts true and false as valid inputs. It activates the pilot signal when true. *Pilot Deviation* is a numeric control that sets the pilot deviation for values 0 kHz-9.9 kHz, in steps of 0.1 kHz.

Along with the RE540's composite stereo signal, an SAP modulator with independent modulation source is incorporated. The *SAP Control* is a boolean that accepts true and false as valid inputs. It activates the *SAP channel* when true. *SAP Carrier*

Deviation Level is a numeric control that accepts values 0.1 kHz-20.0 kHz, in steps of 0.1 kHz. The values define the SAP carrier deviation. The *SAP EQ Level* control accepts inputs 0 %-99.9 % in steps of 0.1% and values of 100 %-150 % in steps of 1.0 %. The RE540 computes the equivalent modulation depth for the SAP channel. *SAP Mod Frequency* is a numeric control that accepts as input values 0.1 kHz-10 kHz in steps of 0.1 kHz. This is the frequency of the SAP channel modulation source. The *S Filter* Control is a boolean that accepts true and false as valid inputs. It activates the internal bandpass filter when true.

The responses for the VI were the Data Command String which contained the data string that was sent to the RE540 via the GPIB. The *GPIB Event Code* has a 0 default setting that indicates that no error has occurred. A *GPIB Event Code* greater than 0 indicated that an error had occurred.

Consider the block diagram for RE540c in Figure 5.6. It is apparent that the construction of the command string that was sent to the RE540 followed a distinct pattern through the use of nested *case structures*, *data range functions*, *PL&A's*, and *concatenaters*. To avoid extreme redundancy in the explanation of the operations of the block diagram, the author has taken liberty with the discussion of code details to be terse but sufficient.

With the exception of the three boolean inputs *Pilot Control*, *SAP Control*, and *S Filter* Control, all of the controls needed data range limiters in order for the RE540 to respond correctly to

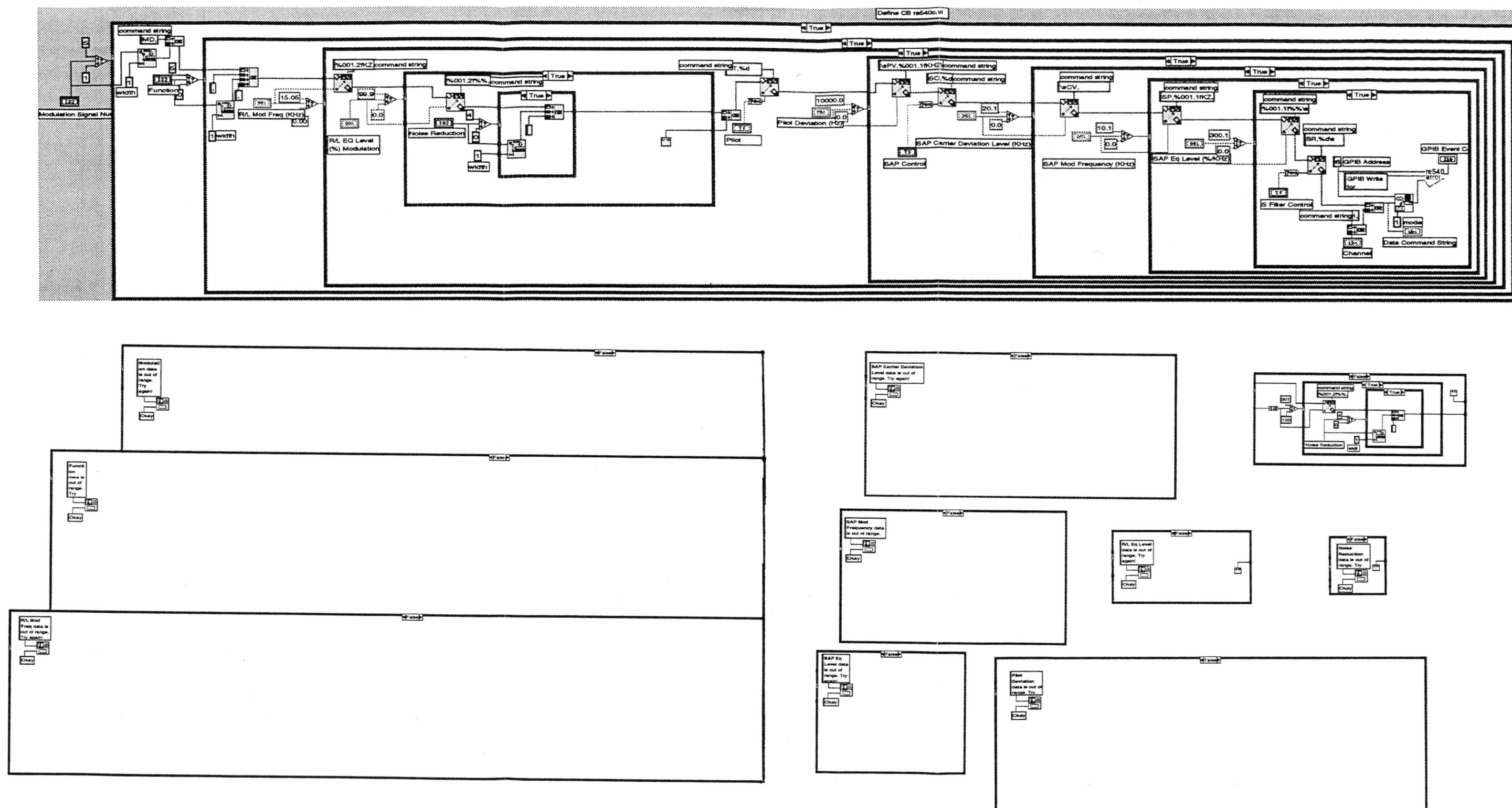


Figure 5.6 Block Diagram of Re540c VI

commands. In the event that an input was out of range, the operator would need notification of the problem. This was handled for each control requiring limiting through use of LabVIEW's *In Range* function. If the data input was out of range a message was sent to a dialog box. Command string construction begins with the *Modulation Signal Number* passing its value to the *In Range* function to check for data validity. The input is restricted to the numbers 1, 2, 3, and 4. The *In Range* function passes true for X inputs that are greater than or equal to the minimum (1) and less than the maximum (5). The case structure receives the output of the *In Range*. If true is passed, then the value is passed to a *To Decimal* function for conversion of an integer to a decimal string. The string value is then passed to the *Concatenate String* function where it is appended to the Modulation Source Definition Command (MD). The first parameter of the MD command is passed forward for continued command building. If the value passed from the *In Range* is false, the false case is invoked. A dialog box is executed. It identifies the input, and states an out-of-range message.

The data range for the next parameter, *Function* control is checked in the exact same manner as previously mentioned. If true is passed from the *In Range* function, the existing partial command is passed to be appended to by the decimal string from the *To Decimal* function. The *Concatenation* function appends the two subsets along with the required delimiters, commas. The output from the *Concatenate String* function is passed forward for further

instructions. The false case for out-of-range *Function* data initiates the dialog box with its message.

Throughout the design *To Decimal* functions were used for integer type inputs, but the *Format and Append (F&A)* function was used for inputs that required floating point data. This is demonstrated for the *R/L Mod Freq*. Valid data are passed to the *F&A* function where they are formatted according to the *F&A*'s format string input. The *F&A* function appends a new formatted string to the existing partial string and passes the entire string. Out-of-range data initiated an error message.

The command continued to be built in the same fashion with two notable changes that are worth mentioning. In the event that a control had different resolutions for different data ranges, as many nested cases as ranges were used to successfully implement the design. The *R/L EQ Level Modulation Depth (R/LMD)* is a good example of this construction where the first data range of 0 -99.9 when in range followed the described pattern; however, when exceeded, the false case led to another *In Range* function where the second range 100-301, was checked. If the second range was exceeded then a *dialog box* supplied the error message, otherwise the process remained intact. The inclusion of the three booleans in the command set was handled by using a boolean to (0,1) function and the *F&A* function. The 0 or 1 was used as the number input of the *F&A* function. The output included the new formatted string appended to the existing partial command. The process continues

as described until true condition of the innermost case structure is encountered. At this point the last command, the channel command is added to the command string via two successive *Concatenate String* functions. The first adds the desired channel to the CH command and delimiter, the second adds the result of the first to the data string. The complete command string is channeled to the *Data Command String* indicator for front panel viewing and the *GPIB Write VI* which sends the command string to the RE540 which is located at address 6 on the bus. The R540errrc error checking and handling VI is also addressed here. The sub-VI reports the error status back to RE540c VI by passing the numeric output to the *GPIB Event Code* indicator.

SMT02

The ATF utilized two Rhode & Schwarz SMT02 generators. These are shown in Figure 5.7. The instruments were named SMT02A and SMT02B. The instruments produced a wide range of signals. These included various test patterns, and modulation schemes. The SMT02's allowed manual operation through its front panel and remote operation via an IEEE-488 GPIB. The GPIB addresses were 7 and 8 for the SMT02A and SMT02B respectively. The drivers followed the ATF instrument driver conventions. Initialization, presetting, resetting, and error reporting routines were designed. Additionally, the documentation conventions for labeling constants, color coding, providing BDL, and descriptions for controls, indicators, icons, and the VI were also adhered to. The

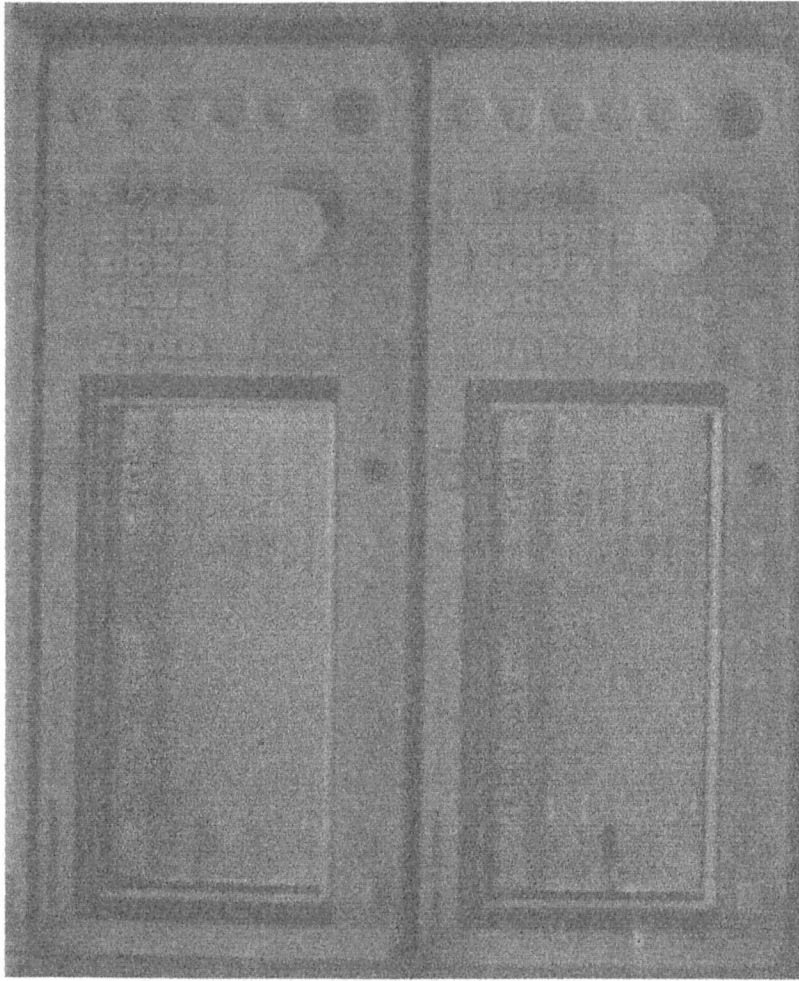


Figure 5.7 Rohde & Schwarz SMT02 Signal Generators

three VI's that were used for controlling the SMT02's are `smterrc.vi`, `smtinitc.vi`, and `smt02c.vi`.

`Smtinitc.vi`

Similarly to the `reinitc` VI, the initialization and resetting were handled by the `smtinitc` VI. It was used to set the SMT02 instruments to a known state. In Figure 5.8, the VI's front panel and block diagram are shown. The VI has one stimulus and one output. The *SMT02 Generator Choice* control allows a choice of the two SMT02's to communicate with. This is done by supplying a 0 or 1 to the control's connector terminal. The control is a text ring control that is connected to the selector on a case structure. If the SMT02A is chosen a 0 is sent which causes a string constant 7 to pass out as the address. However, if the SMT02B is chosen a 1 is sent which causes a string constant 8 to pass out as the address. The *GPIB Event Code* is the error reporting indicator that is passed from the error checking sub-VI (`smterrc`). The initialization VI uses the *GPIB Write* VI to send the instruction set to the chosen SMT02. The *GPIB Write* VI accepts at its input ports the: address string (7 or 8 for the SMT02's) as denoted by the pink wire and pink string constant box; data string that contains the device's instruction set and mode of termination for the *GPIB Write* which is marked as an integer 2 for this device. The 2 calls for the *GPIB Write* VI to append a line feed to the string and to send an EOI with it. The data that are sent to the SMT02 instruct it to generate a signal with a center frequency of 39.750 MHz and set the

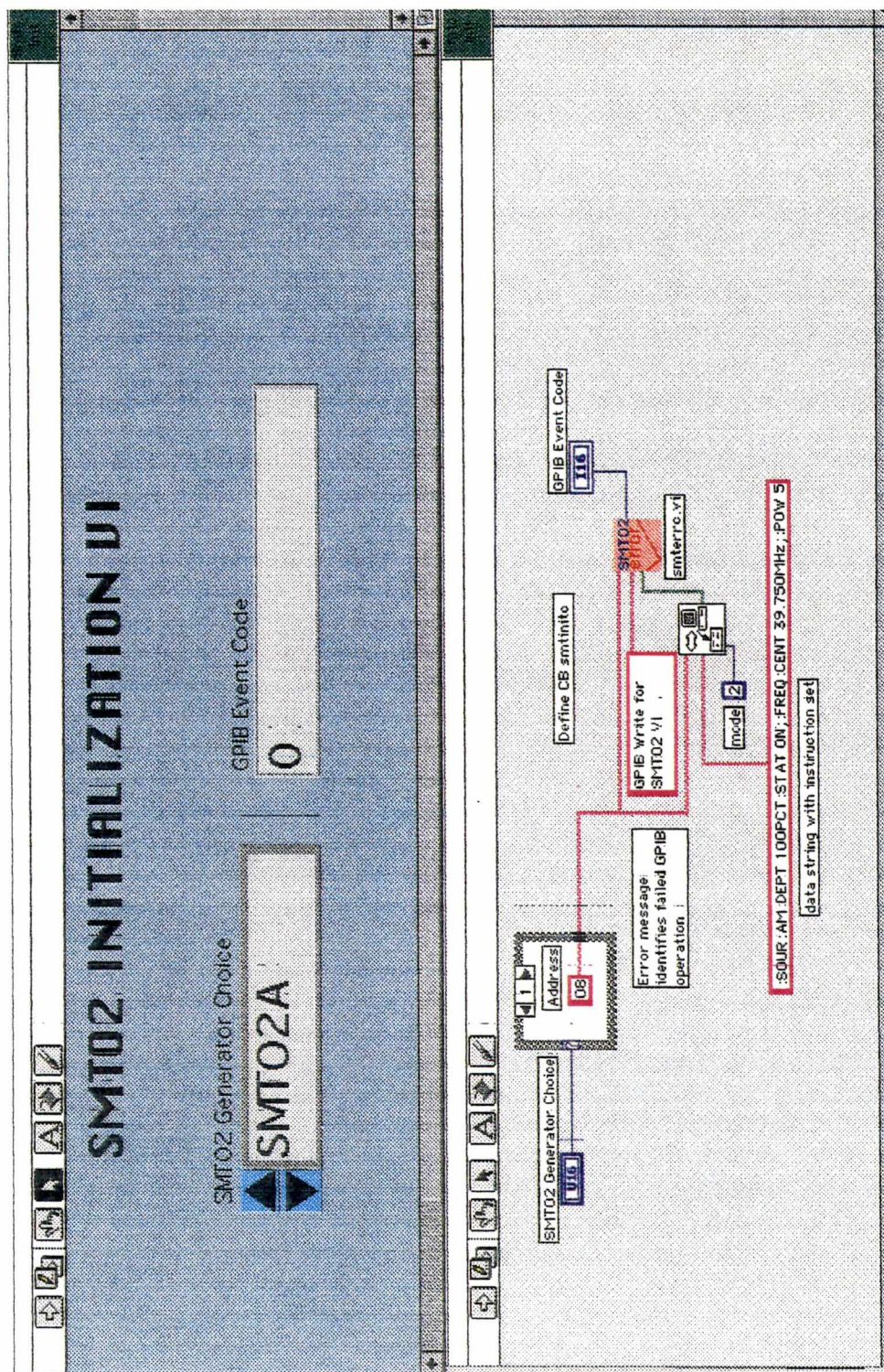


Figure 5.8 Front Panel and Block Diagram of Smtinitc VI

amplitude modulation (AM) to 100% for it. The signal strength is then reduced by 5 DB. The *GPiB Write* VI returns a 16-bit status boolean array where each bit defines a state of the GPIB Controller. Bit 15 is set when an error occurs. The status is sent to the error checker, *smterrc.vi*.

Smterrc.vi

Like the *r540errc* error checking VI, the *smterrc* VI was constructed from the TSI-8150 prototype. The generic error checking VI was customized for the SMT02 by changing the name of the instrument in the error message header.

Figures 5.9 and 5.10 show the VI's front panel and block diagram respectively. The design and functionality of the VI mirror that of the *r540errc* VI. To avoid redundancy, the author refers the reader to the *r540errc* section on pages (77-82) where a thorough discussion of the functions of the VI is provided.

Smt02c.vi

The *Smt02c* front panel is shown in Figure 5.11. The SMT02 *Generator Choice* control allows a choice of the two SMT02's to communicate with. An operator supplies the frequency of the interference signal to be generated to the Interference Signal Freq control. The *Attenuation* control allows the output signal level to be controlled. The operator enters the amount of suppression or amplification in the range of 144 dbmv-16 dbmv. The *Percent Amplitude Modulation* control accepts inputs of 0 %-100 %..

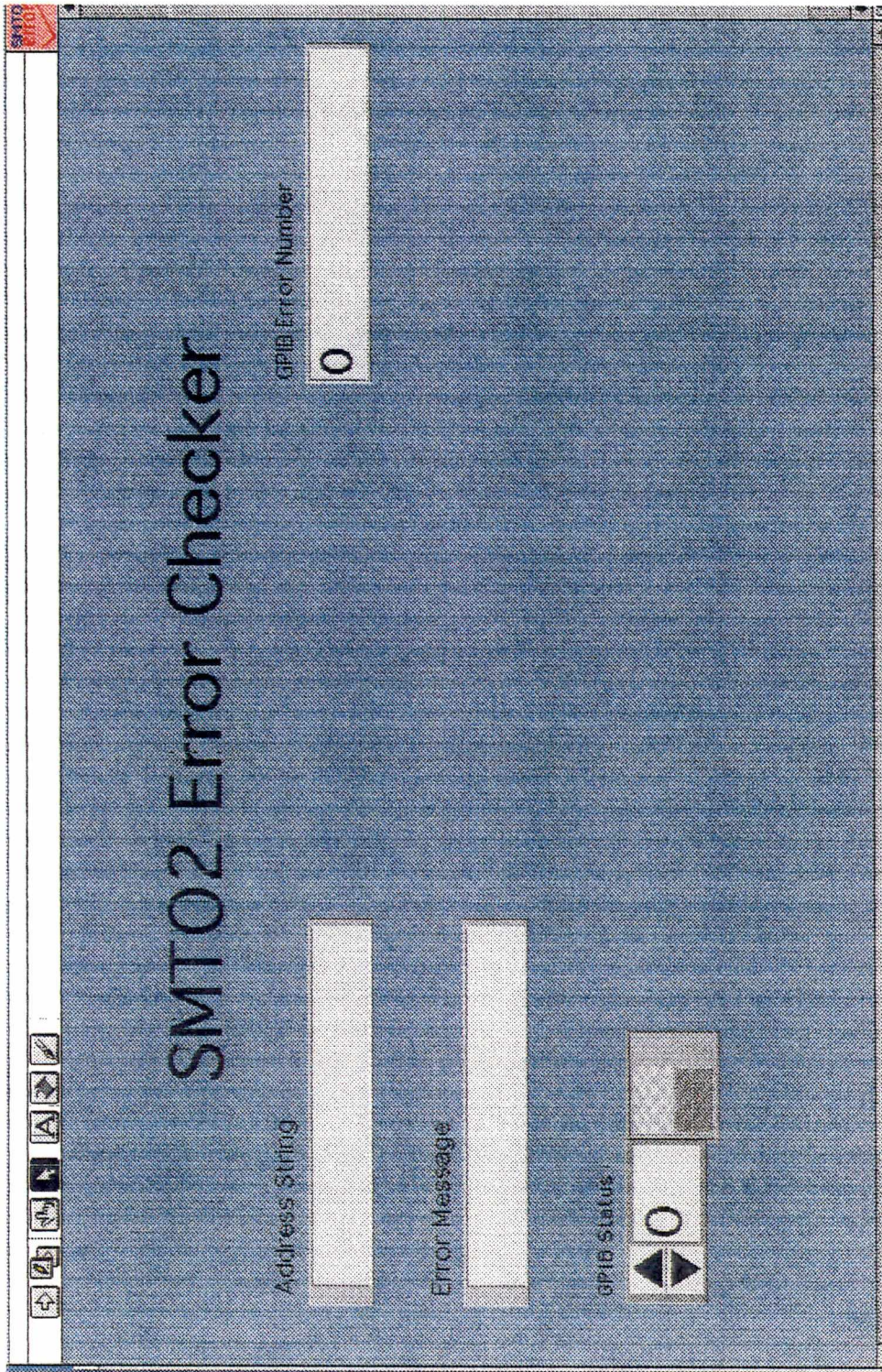


Figure 5.9 Front Panel of Smterrc VI

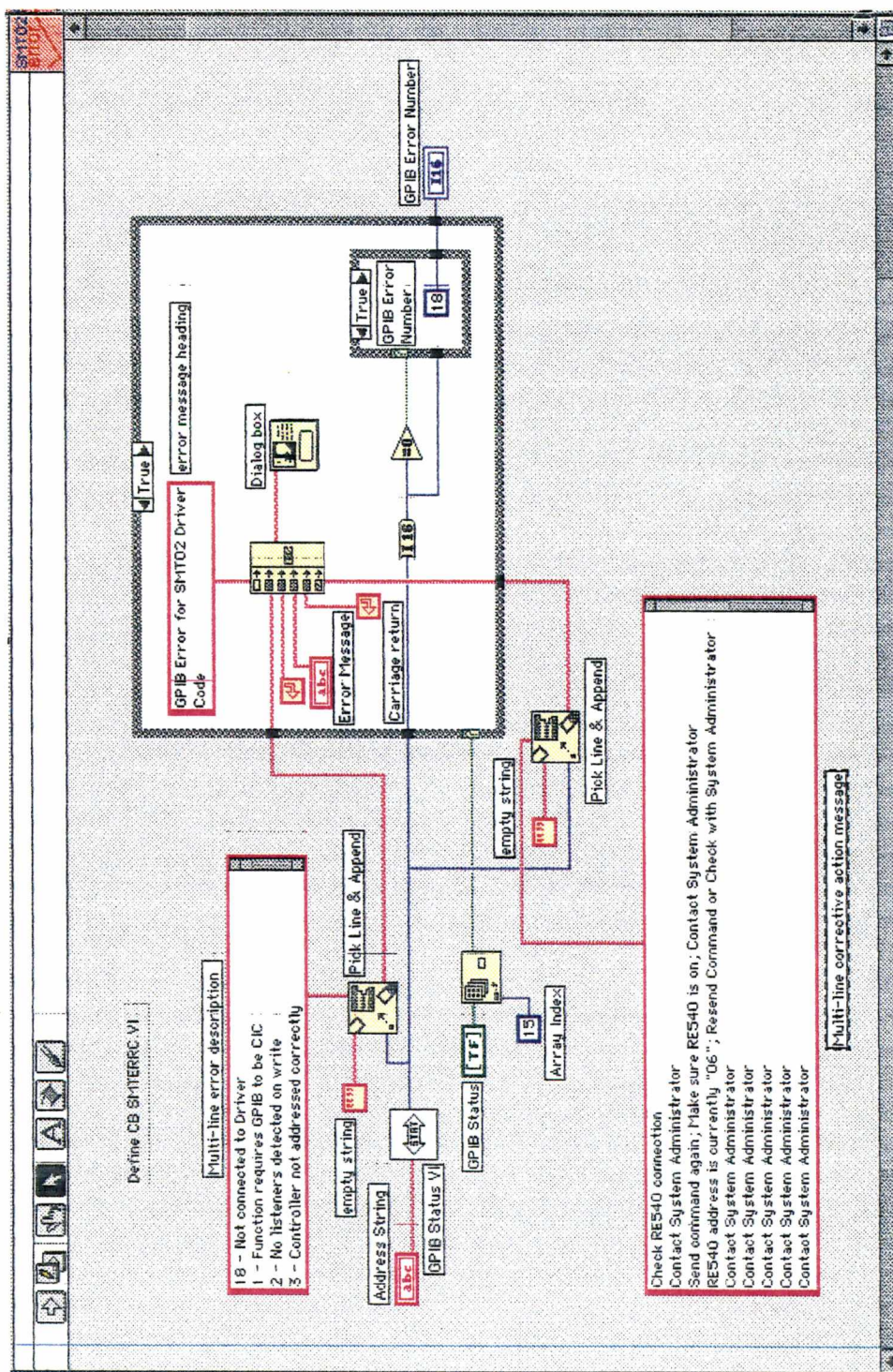


Figure 5.10 Block Diagram of Smterrc VI

It controls the AM modulation depth. The frequency of the primary signal to be generated is supplied to the *Percent Signal Frequency* control. The *Test Selector Text Ring* allows the choice of set-ups for the five tests that use the SMT02. The selected option causes a corresponding ring number to be passed from its block diagram terminal. Finally, the sole output is the *GPIB Event Code* that indicates bus errors by the number returned from the error checking VI.

The *Smt02c* block diagram is pictured in Figure 5.12 and Figure 5.13. The *Percent AM* and *Attenuation* values are each changed to strings by the *To Decimal* function. Similarly, the *Interference Signal Freq* and *Primary Signal Frequency* values are changed to strings through the use of the *F&A* function. The Test Selector control terminal is passed a number that corresponds to the selected option. The number 0 corresponds to the Do Nothing alternative and invokes case 0 of the case structure where an empty string is forwarded from the structure's terminal. Each of the other five cases corresponds to the ordered elements of the *Test Selector* ring: IF Reject corresponds to case 1; CB Reject corresponds to case 2; FM Reject corresponds to case 3; 920 Reject corresponds to case 4; and AFT- Pull In corresponds to case 5. Each of the cases utilizes series of string constants that contain the SMT02 commands, a *Concatenate String* function to tie in the appropriate values from the controls and to combine the individual commands into a single command line. The command set for cases 1-3 are identical.

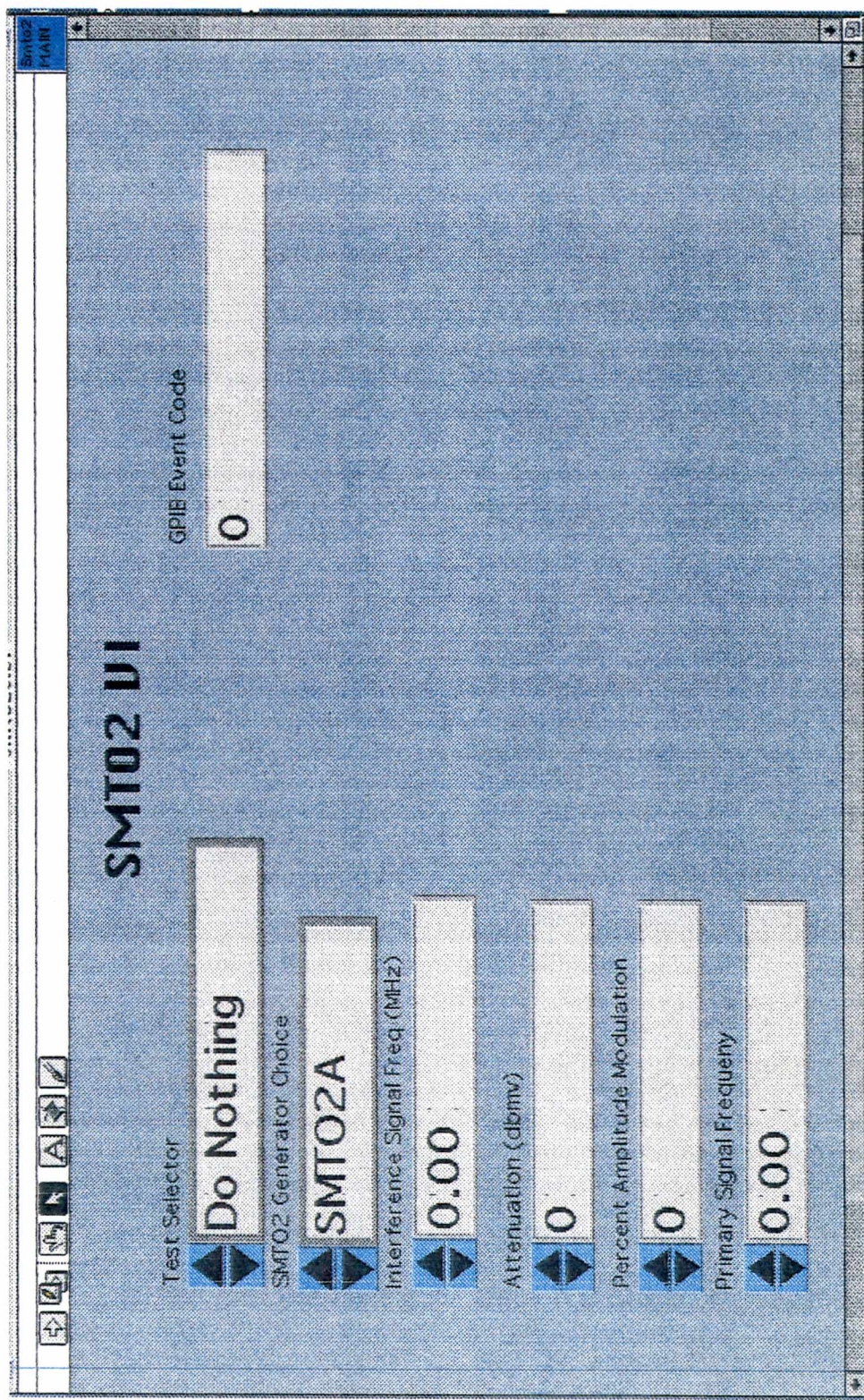


Figure 5.11 Front Panel of Smt02c VI

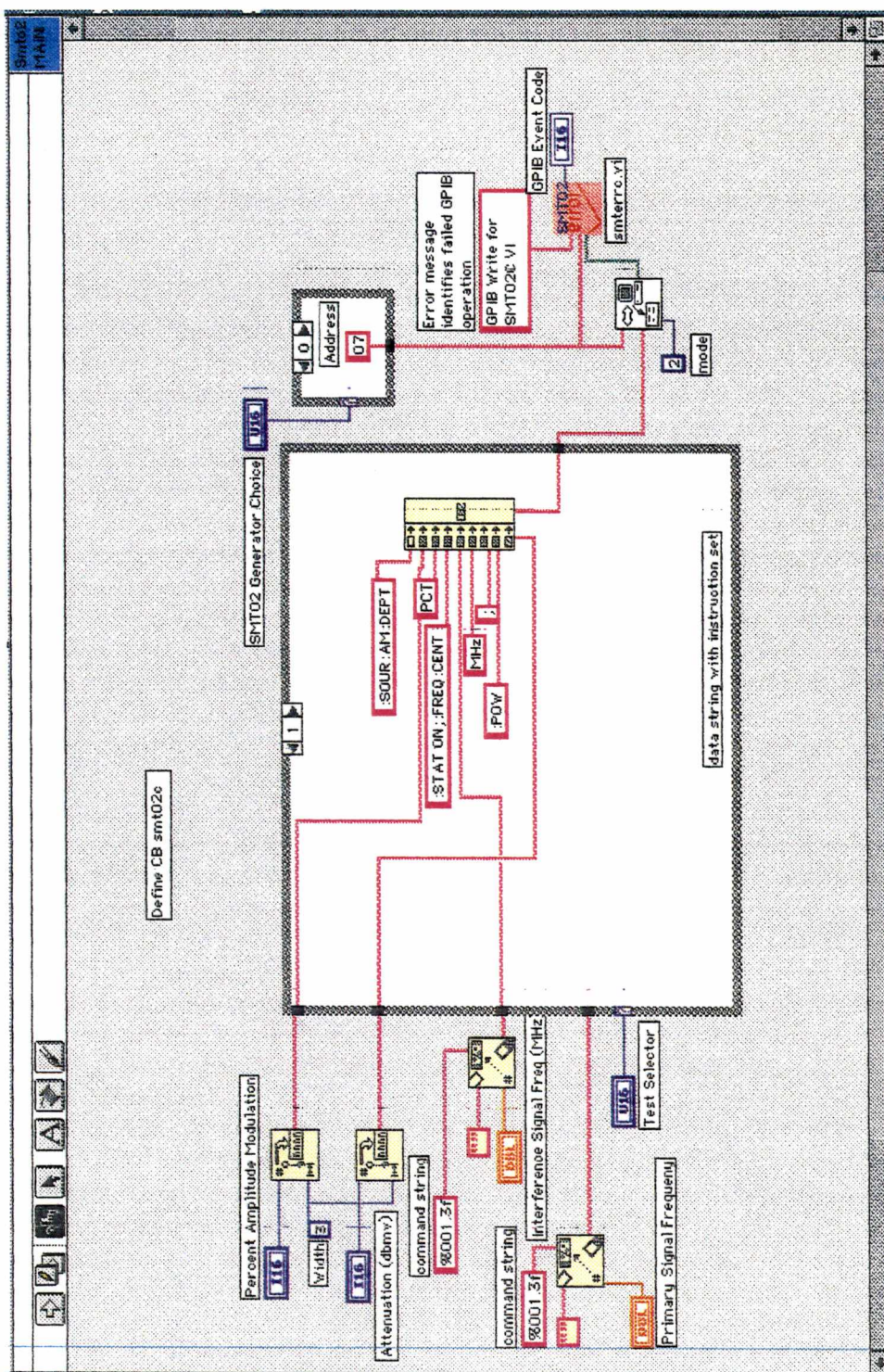


Figure 5.12 Block Diagram of Smt02c VI

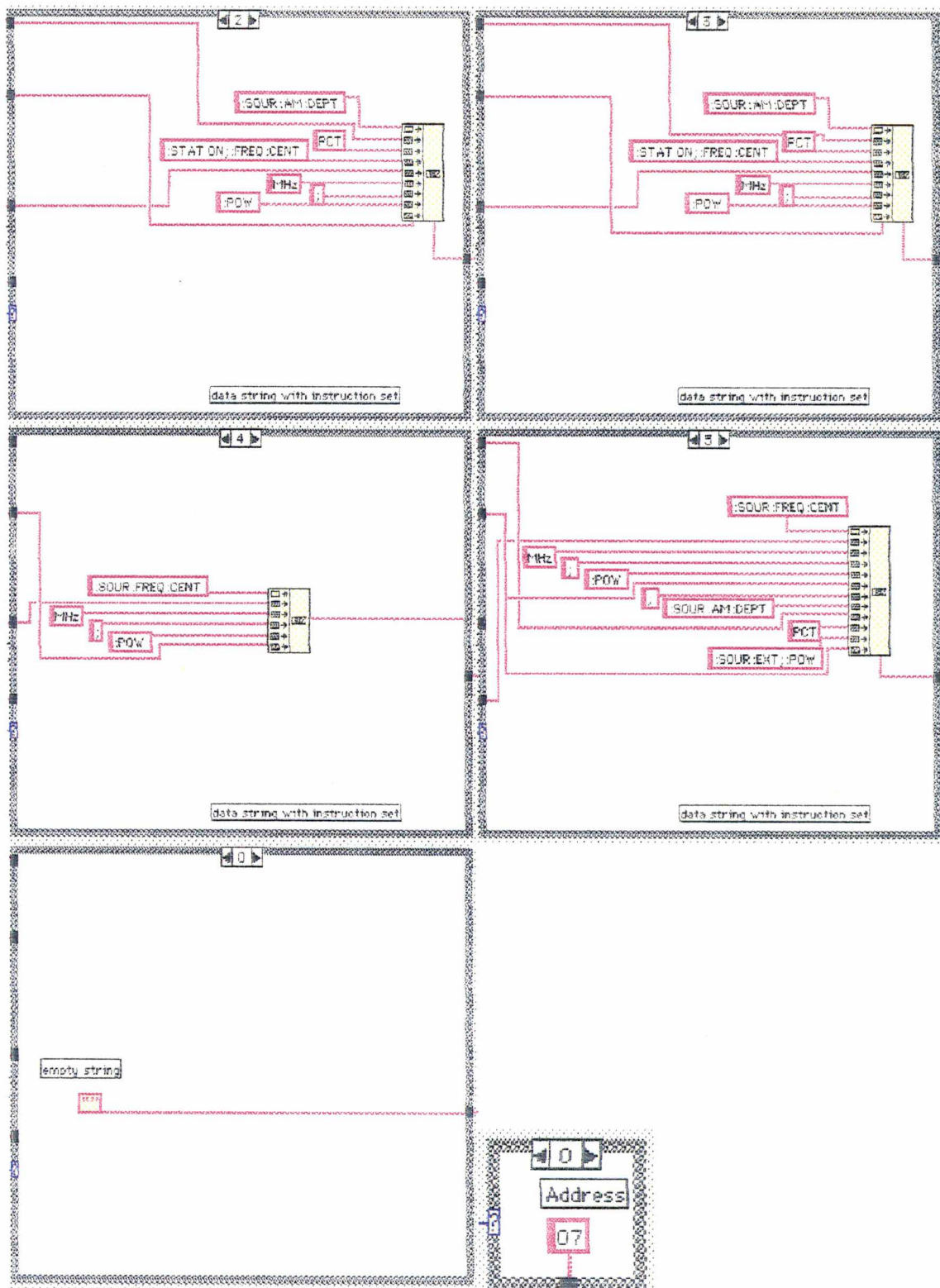


Figure 5.13 Block Diagram for Smt02c VI

The values from the *Percent AM*, *Attenuation*, and *Interference Signal Freq* controls are passed to the *Concatenate String* function. The *Percent AM* value is appended after the string constant for AM depth defining. The percent command is concatenated next. The interference signal command, value, and units in MHz are added next. The string is then appended with a command separator and the command for signal level control. The last addition is the attenuation value. The complete command is passed to the structure's terminal. In similar fashion the fourth and fifth cases are constructed. From the Case terminal, the command is passed to the *GPIB Write VI* to send the instruction set to the chosen SMT02. If the SMT02A is chosen, a 0 is sent from the SMT02 Generator control to the selector at the Case's terminal that causes a string constant 7 to pass as the address. On the other hand, if the SMT02B is chosen, a 1 is sent which causes a string constant 8 to pass out as the address. The *GPIB Write VI* accepts at its input ports the: address string; data string that contains the device's instruction set; and the mode of termination for the *GPIB Write* that is marked as an integer (2) for this device. The *GPIB Write VI* returns a 16-bit status boolean array where each bit defines a state of the GPIB Controller. Bit 15 is set when an error occurs. The status is sent to the error checker, *smterrc.vi*. If no error is found, the *GPIB Event Code* is returned a 0; alternately a numeric that corresponds to an error description is returned in the case of an error.

RF Overload

The structure of the incremental development plan was such that the author had developed or co-developed drivers for four instruments prior to completion of the development of a test. Tests were scheduled according to their similarity to the chosen test prototype. The tests that were most similar were scheduled in the earliest increments possible. This criterion was chosen because it affected the scheduling of instrument driver development. The approach allowed test code and instrument drivers to be developed in parallel.

A fellow team member was tasked with the development of the project's prototype test, RF Sensitivity [34]. Of the seven or so tests that were determined to be similar enough to warrant inclusion in one of the earlier increments, the RF Overload test was one.

The purpose of the RF Overload test is to determine and measure the level at which a television receiver could no longer receive a signal without noticeable distortion due to an excessive RF level input. The test requires the test operator to carefully observe the behavior of an NTSC color bar signal on VHF channels 5 and 13. The monitoring of the set's responses to level increases leads to the determination of an RF level where the distortion is just perceptible. Admittedly, it takes a trained and skilled eye to discern such a point. Due to many spurious signals, to successfully execute this test, it must be conducted in an RF free area. Recall that an RF free

area allows suppression of undesired RF signals. The RF Overload test procedure is outlined in Figure 5.14. Its hardware connectivity scheme is depicted in Figure 5.15. It's not suprising that manual execution of the RF Overload test requires a significant time investment to set-up equipment and configure devices. Such conditions made the the RF Overload test an excellent candidate for automating.

Test Panel

Code developement conventions were mentioned earlier. The conventions for instrument drivers were considerably more relaxed than the conventions for the development of tests. Understandably so, since unlike the instrument drivers, each test panel served as the user interface to it's respective test environment. It was of utmost importance that the user interface be consistent aesthetically and functionally, professional, and pleasing. The prototype was developed as a generic test template that was made up of a general front panel layout and logic blocks common to all tests [34].

The team adopted the front panel as shown in Figure 5.16. The test name by convention is the topmost item on the panel. The large green square contains generic controls for measuring and recording measurements. It contains a vertical slide control, *Relative Attenuation*. The control allows the selection of the amount of attenuation to apply to the test signal.

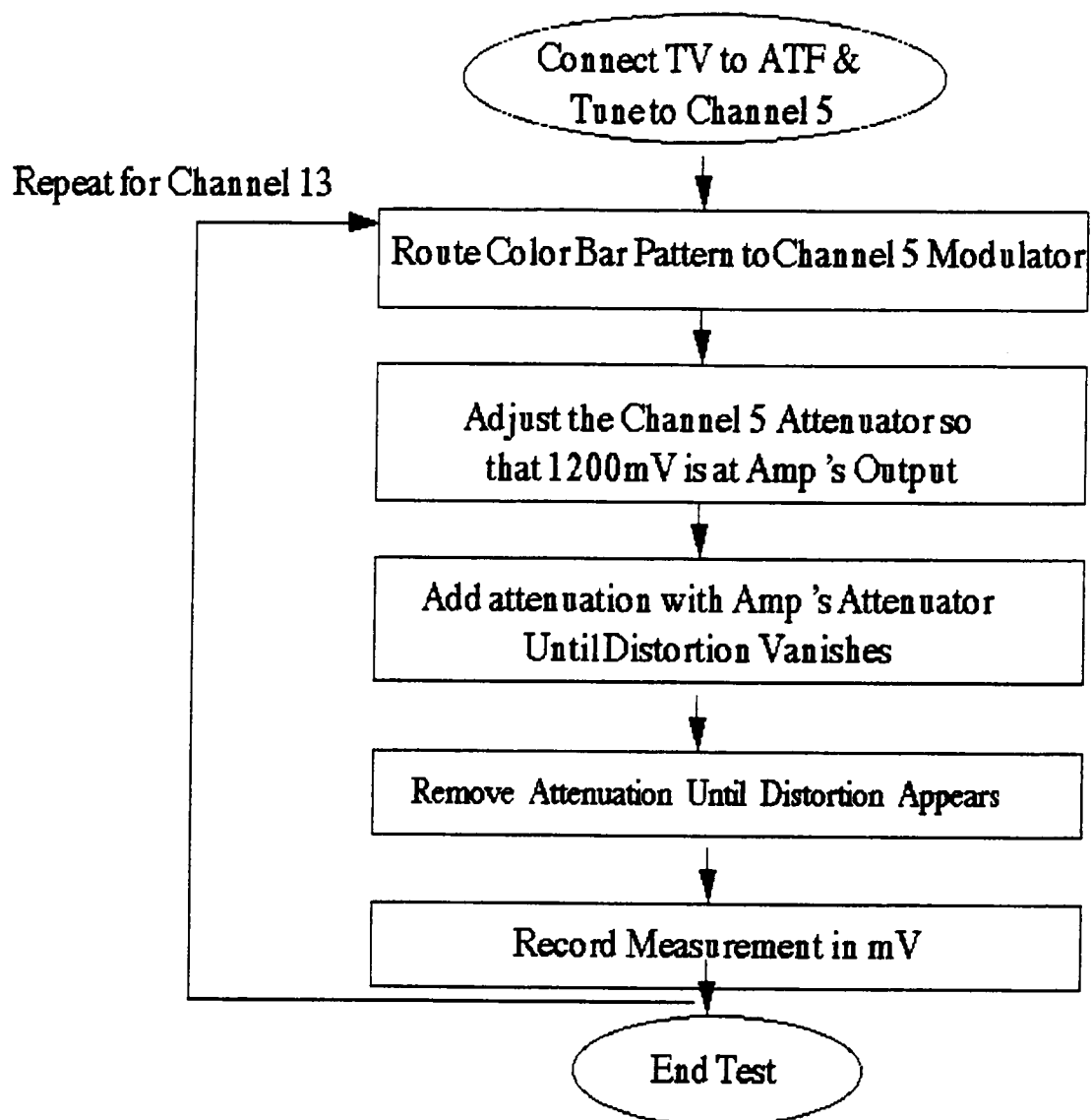


Figure 5.14 RF Overload Skeletal Test Procedure

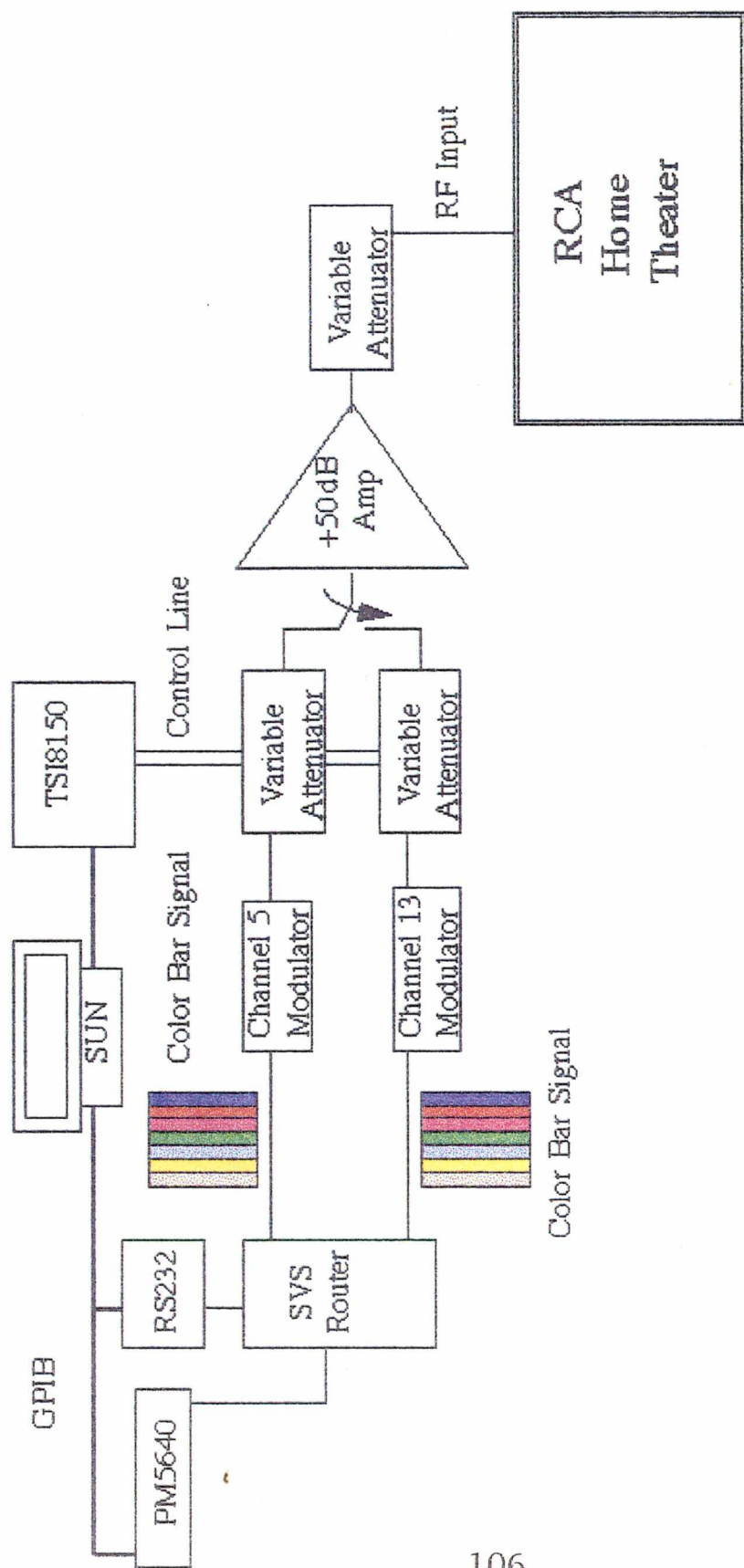


Figure 5.15 RF Overload Interconnectivity Diagram

RF Overload

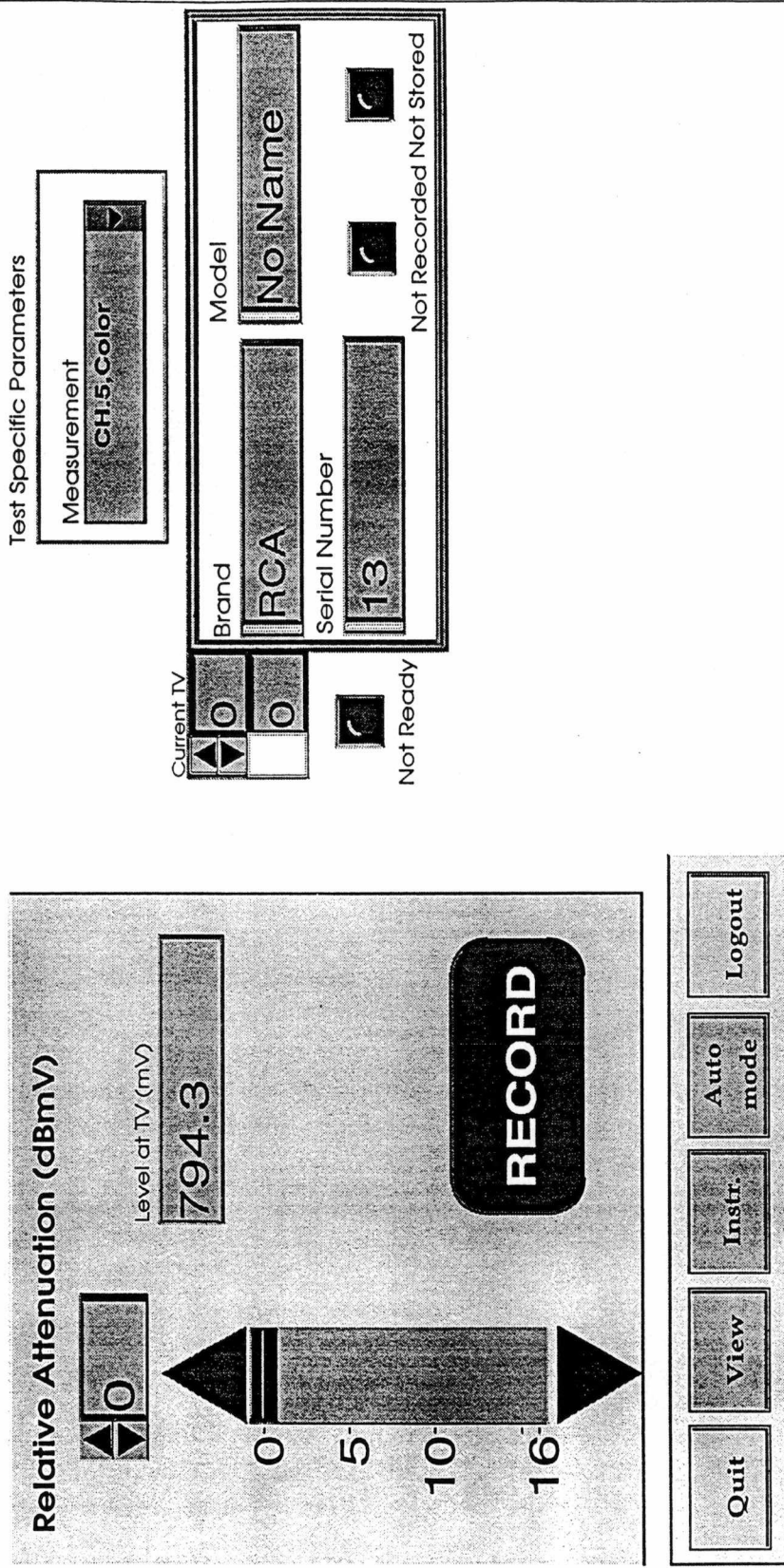


Figure 5.16 RF Overload Front Panel

The desired amount is sent to the TSI8150 for attenuator control. The amount of attenuation is also displayed in the numeric indicator above the slider. The input RF level (dBmv) that correlates with the applied attenuation is shown in the *Level at TV* indicator. The *Record* button is a boolean control that allows a measurement, its parameters, units, and other pertinent information regarding the television under test to be stored and viewed via a lower level panel. The section below the green square is home to various boolean test options that are available to the operator. All of the selections are shown in their default state. The *Quit* button ceases test execution and transfers execution to a higher level panel, the RF Menu. The *View* button allows the user to see measurements that have been recorded. These are shown on a lower level panel. There is also a set of instructions that explain the test's purpose and functionality. The *Instruction* panel appears when the *Instr* button is depressed. The test mode, automatic or manual is determined by the *Auto Mode* button. *Auto Mode* allows the test to proceed in the default auto-sequencing mode. Manual mode lets the user choose the order in which television sets are tested or the order in which test parameters are applied. The *Logout* option stops test execution and passes control to the Category Menu, a higher level menu. The right side of the panel is headed by the test specific parameter, the *Measurement* ring control. The control displays the current test parameter that is being measured. The *Current TV* cluster indicates the brand, serial number, and model of the current television under test. The information is passed from the higher

level TV header where it is entered. The TV Header is shown in Figure 5.17 [33]. There are two boolean indicators located in the cluster also. They show the recorded and saved status of each measurement respectively. The red color indicates that the measurement has not been recorded. If the measurement is recorded, the indicator becomes green. Similarly, the storage indicator is green for data that is stored and red otherwise. The test uses another indicator to show the status of the current television. If 15 minutes have elapsed since the television's information was entered in the header, the indicator changes from its default red color to green.

Block Diagram

The test template consisted of five distinct stages, each contained within its respective frame of a sequence structure. The first stage is the BIT Checking stage. The BIT Checking Stages frame is shown in Figure 5.18. The `bit_chek` VI serves as an interface to the BIT routines and passes a boolean result to indicate pass (true) or fail (false) [34]. The test proceeds if true is returned; on the other hand, a returned false indicates that the BIT located a terminal error and the test is aborted. This stage also dims out all front panel controls that are located on the green pad and the test mode option through the use of attribute nodes.

Sequence frame one houses the Initialization stage. It is shown in Figure 5.19. This stage was used for setting the initial

TV Header List

TV Header

 0

Brand	Magnavox
Model	108hd56
Screen Size	20
Version	145ed
Serial Number	765234
Outlet	32

Clear

Add

Delete

Done

Figure 5.17 TV Header

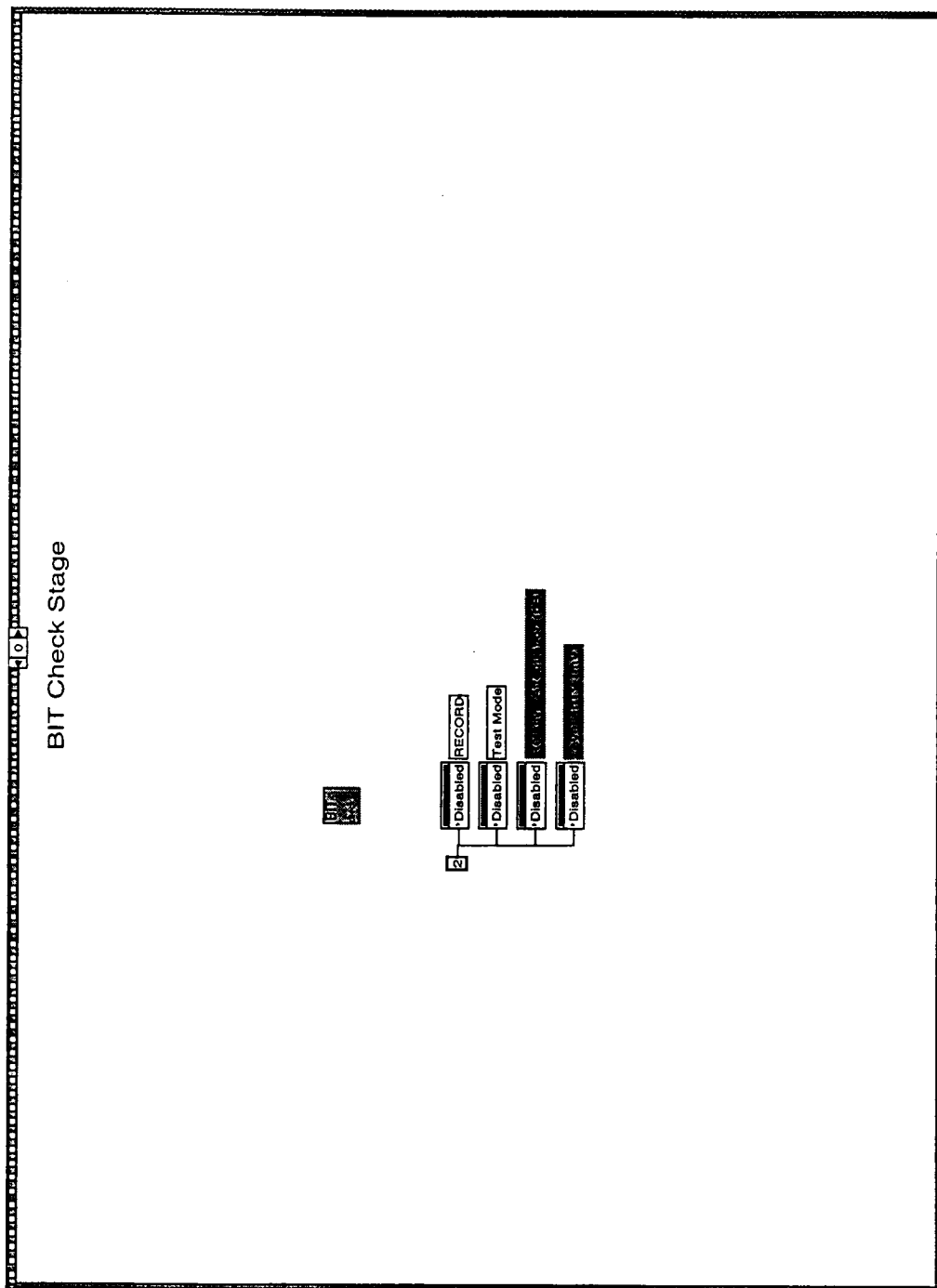


Figure 5.18 RF Overload Block Diagram-BIT Stage

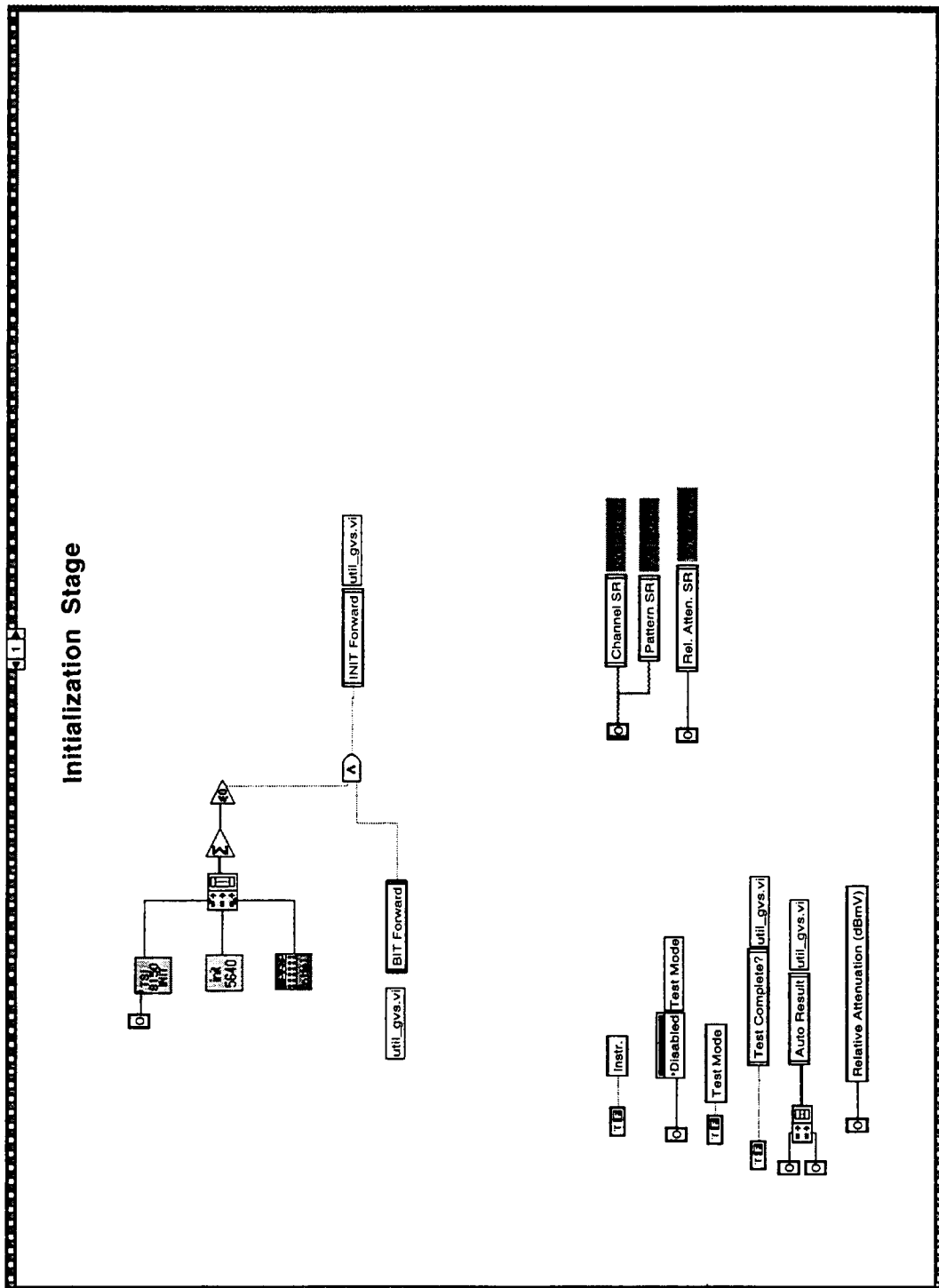


Figure 5.19 RF Overload Block Diagram-Initialization Stage

states for the instruments that were to be used in the test. The *TSIinit* VI sent a 0 to instruct the TSI8150 to close all the switches [33]. The *Init5640* VI set the PM5640 to the initial state as set by the VI's designer [35]. Similarly, the *SVSinit* VI sets the Sierra Router to a known state and clears all previous connections [36]. Each of the VI's outputs, an *Event code* is collected in an array function and summed. Next, the result is compared to 0. The output is passed to an *AND* function with the forwarded result from the BIT check routine to the *Init Forward* global variable. Since 0 was by convention the event code to signify that no error had been found, if the error indicator was greater than 0 it would cause the test to cease. Several local variables were also initialized in this stage. boolean constants initialize the *Inst*, *Test Mode*, and *Test Complete* controls. An attribute node is used to enable the Test Mode. A local variable, *Relative Attenuation* and global variables *Channel SR*, *Pattern SR*, *Rel. Atten. SR*, and *Auto Result* are all initialized to 0 [34].

The Preset stage allowed test conditions to be set prior to testing. This stage is depicted in Figure 5.20. The *TSIomni* VI used to attenuate the channel 5 signal by 4 dBmv. The GPIB address, 2 was used to send the instructions to the TSI8150 [33]. The VI passes as output, the GPIB error number which is sent to a *Build Array* function. The *PM5640* VI is used to initiate a color bar signal from the PM5640 [35]. The *SVSxcon* VI accepts the *Cross Connect* global variable string cluster as input. The global variable defines the output, input, and type of signal to be routed. The VI

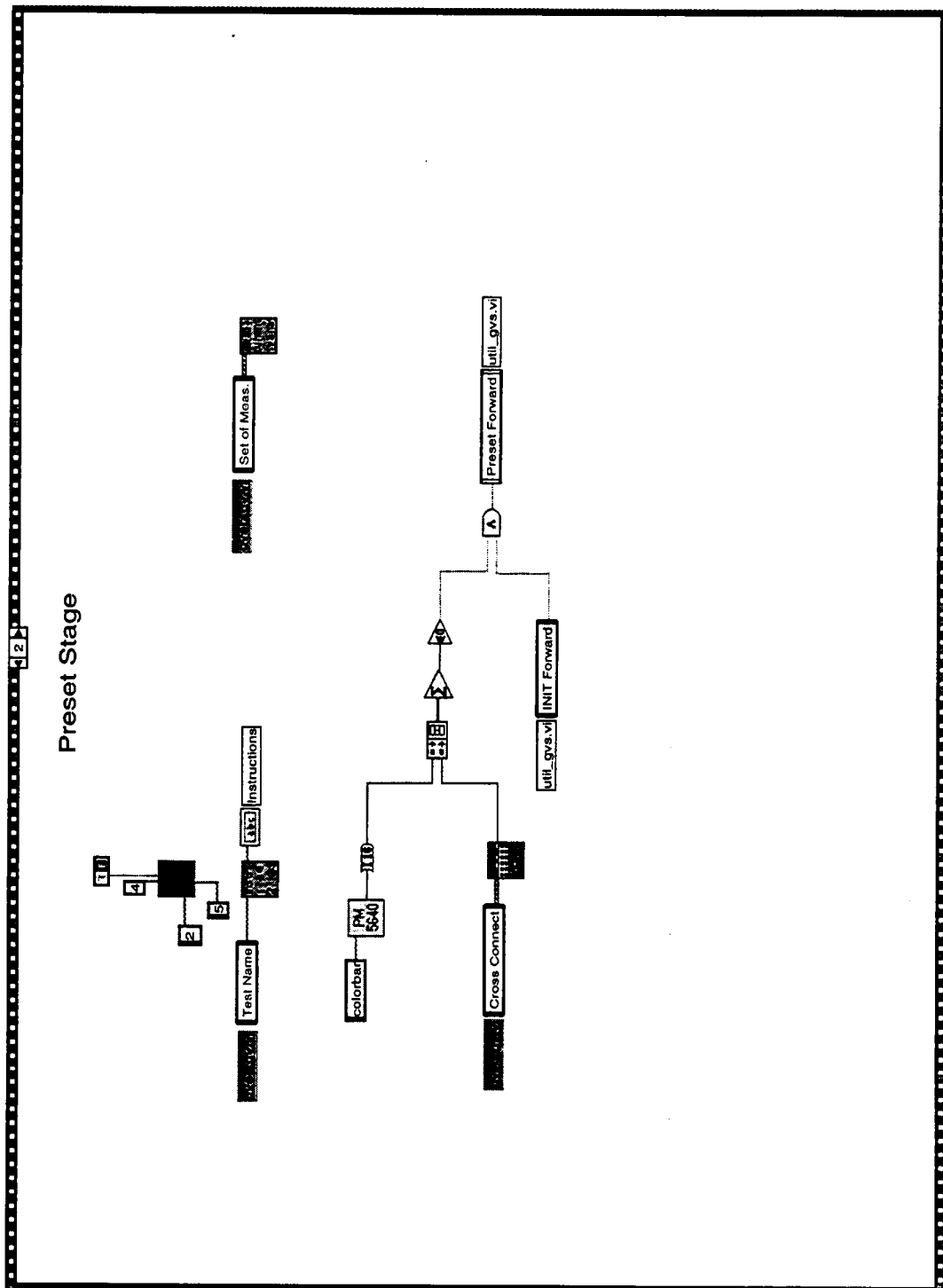


Figure 5.20 RF Overload Block Diagram-Preset Stage

makes connections to the SVS Router's output from the input and level combinations as specified by the global variable [36]. Each of the VIs output an *Event Code* which was collected in an *Array* function and summed. The output of the sum was compared to 0. The output passed to an *AND* function with the result forwarded with the *Init forward* result to the *Preset Forward* global variable. If the error indicator is greater than 0 it causes the test to cease. The *Set of Meas.* global variable defines the measurement parameters and units. The array information is passed to the *ini_2d* VI to initialize the two-dimensional data array that is used for tracking all measurements. The *Test name* global variable provides the filename over to the *Instpanel* VI which displays the test instructions for the RF Overload test. The VI outputs the instructions to the instructions indicator [34].

The Test stage contains the logic for measuring, processing, recording, viewing and saving the data to disk. Referring to Figure 5.21, the terminal of the *Instr* button is wired to an *attribute node* that allows the *Instructions* to be displayed when the *Instr* button is true. The *Index Value* attribute node of the *Current TV* indicator supplies the *Warmchek* VI with a 1 dimensional array of the set of television index values [36]. The VI uses the index values to determine if the UUT has warmed for the suggested 15 minutes. Its output is wired to the terminal of the *Warm-up* indicator which changes from red to green when true to indicate that the time criterion has been met. The values stored in the global variable *Current TV* are passed to the indicator *Current TV* to display the

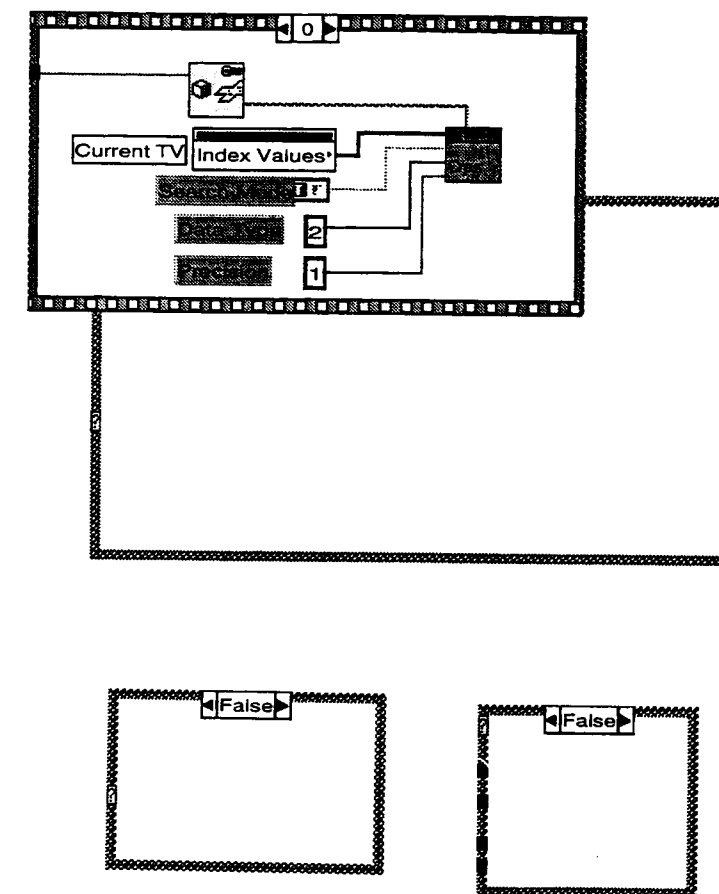
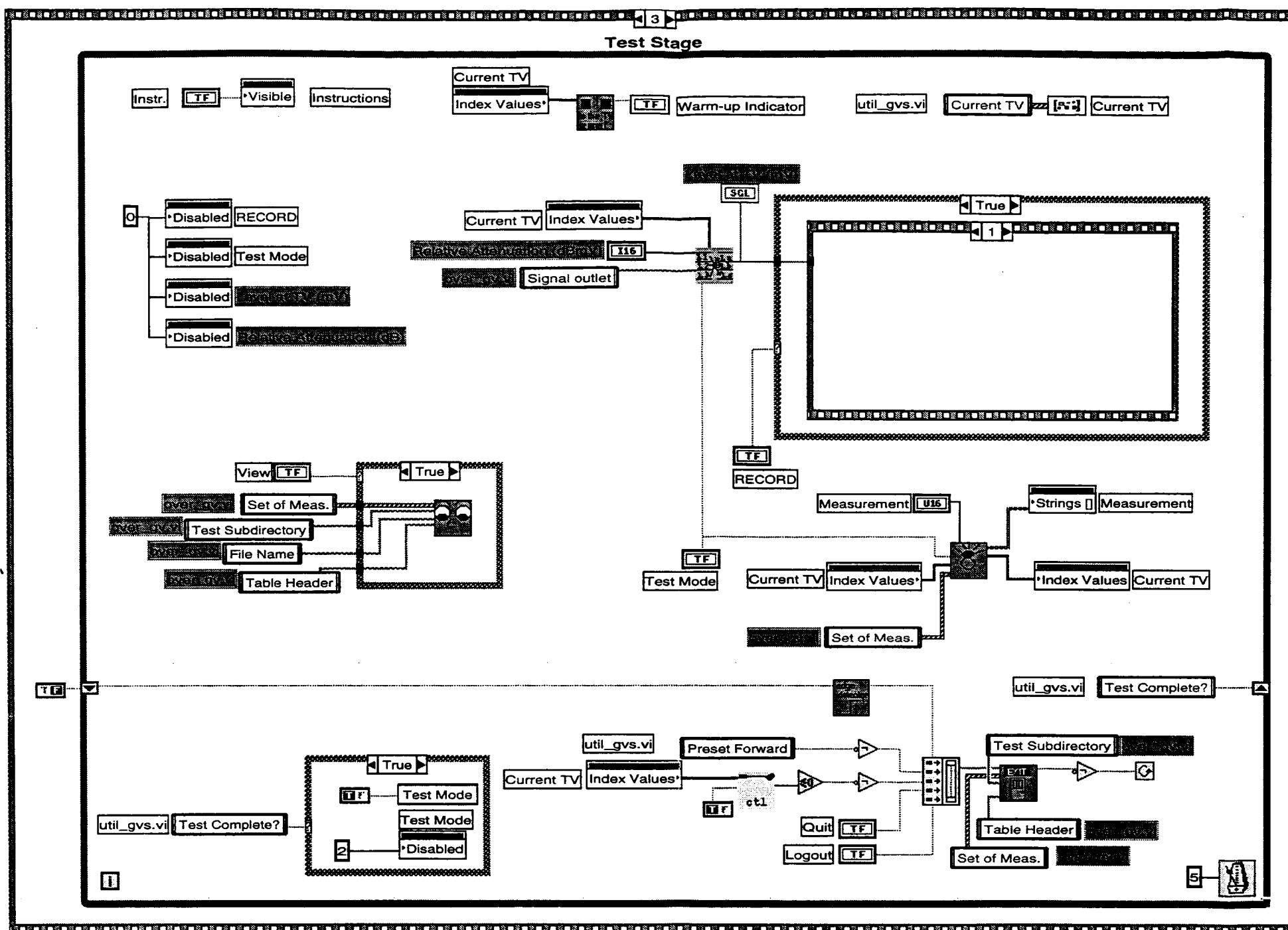


Figure 5.21 RF Overload Block Diagram-Test Stage

values on the VI's front panel. The front panel controls that are located on the green pad and the test mode are all enabled through the use of attribute nodes. The Newxlatr VI accepts as inputs the current television index, the relative attenuation as input by the operator, the *Test Mode*, and the signal outlet which allows manual entry of the outlet number. The VI sets the slider 0 dBmv attenuation level to correspond to the desired starting RF level of 1259 mV. The *Test Mode* allows manual setting of the signal as opposed to receiving the value from the TV Header VI. The VI returns the RF level in mV at the televisions input to the *Level at TV* indicator to be displayed at the front panel. The output is wired to the *case structure* where it is passed for the true case if the *Record* button is selected. The true case uses a *sequence structure* for forcing data dependency. The RF level is passed to a *Flatten to String* function for conversion to a binary string. The string is sent to the Rec_ops VI as the measurement value along with the current television's index, a false for the *Search Mode* boolean, a single *data type* with a precision of 0 [34]. The VI sets the *Data Recorded* flag to true for the current measurement and determines the next step in the auto sequencing. The false case of the structure has no logic. The *View* button's terminal is wired to a case structure. When selected, the true case is invoked which passes the *Set of Meas.*, *Test Subdirectory*, *File Name*, and *Table Header* global variables to the VW_lorml VI [36]. The VI formats and displays the measurement data on the View panel. The View panel also allows the data to printed and saved to the *File Name* in the *Test*

Subdirectory. The View panel is shown in Figure 5.22 [36]. The false case has no logic. The *Mode_ctl* VI was used to control front panel controls and indicators when manual mode is selected [34]. In automatic mode, the controls and indicators are controlled programmatically. The VI accepts the *Measurement* ring control, the *Test Mode*, *Current TV* index values, and the *Set of Measurements*. The *Measurementstring* array and the updated *Current TV Index Values* are the VI's outputs. The *Test Complete* global variable is wired to a case structure where the false case does nothing, but the true case initializes a *Test Mode* local variable and disables and greys out the front panel *Test Mode* control. The *Newdrivr* VI accepts the *Current TV Index Value* and a boolean constant that is set to true [35]. The VI houses all the instrument drivers that are needed for conducting the test. The VI sends test parameters to the the SVS Router and TSI8150 . The VI exits the Event Code which is compared to 0 and passed forward for test termination processing. There is also a *Test Complete* boolean *shift register* set that is initialized to false and passed to the *Edge_det* VI [34]. The VI checks the *Test Complete* global variable status and passes out the Complete indicator as true if the test is complete. The *Complete* indicator along with the *Preset Forward* global Variable, the *Newdrivr* VI *error output*, the *Quit* terminal, and the *Logout* terminal are passed to a *Build Array* function. The *array*, *Table Header*, and *Set of Meas* global variables are sent to the *Exitckek* VI which checks exit conditions and exits if an error was found in the BIT routine, the *Quit* or *Logout* button was selected,

or an error in testing was found. The result is passed to a *Not* function and onto the *while loop* terminator which stops when false is received [34]. Each iteration of the loop takes place in 5 millisecond intervals as defined by the *Wait Until Next Multiple millisecond* function.

The final stage is for resetting. It utilizes the *TSIinit* VI to open all the switches. The constant 1 sets the VI's internal ring control to open all switches [33]. The *Init Equip* global variable contains the reference name for all of the instruments that were used for initialization and resets the devices to their respective default initial states. It supplies the *Dyn_rest* VI with the equipment list. The VI then loads the initialization VI's for execution. Finally, the *Test mode* option and all of the controls that reside on the green pad are disabled and greyed out. The reset stage is shown in Figure 5.23.

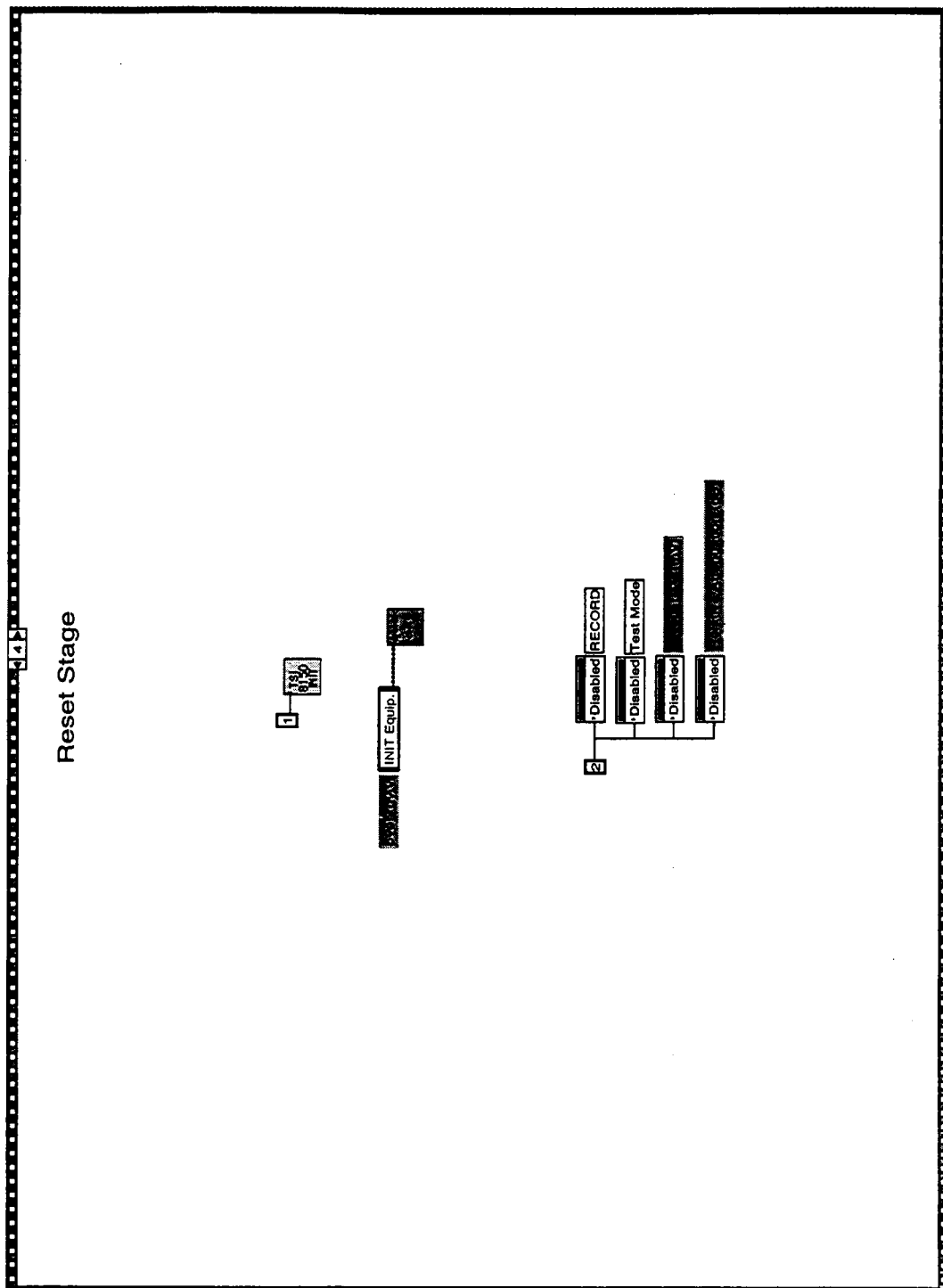


Figure 5.23 RF Overload Block Diagram-Reset Stage

Chapter 6

CONCLUSION

As mentioned in Chapter 3, most cases of ATF implementation have shown that ATEs execute faster and cost less on a per-test basis than do manual test systems. Preliminary results comparing test times for manual implementation to those executed automatically by the ATF system support the aforementioned trend. Consider the data in Figure 6.1. For the tests specified in the legend, the graph compares the estimated manual test times to actual results from executing the tests using the ATF. The estimated values were supplied by a test engineer that is affiliated with the project sponsor. The manual test times include the time for the set-up of the testing system. For example, test equipment, preparation of the televisions for testing, and the test execution time. The manual test system has considerable equipment set-up time associated with it. Since, equipment set-up is done automatically with the ATF, it results in a substantial time saving over manual testing. Figure 6.2 shows the composition of the ATF test execution times. The graph exposes an opportunity for further improvement through efficiency gains. The required television warm-up time and time for entering TV header information account for approximately 20 minutes of an ATF test routine. Obviously, creative methods can be employed to reduce the overhead. Figure 6.3 shows the average test times for manual and automatic testing. Manual testing averaged just over 3 hours at 86

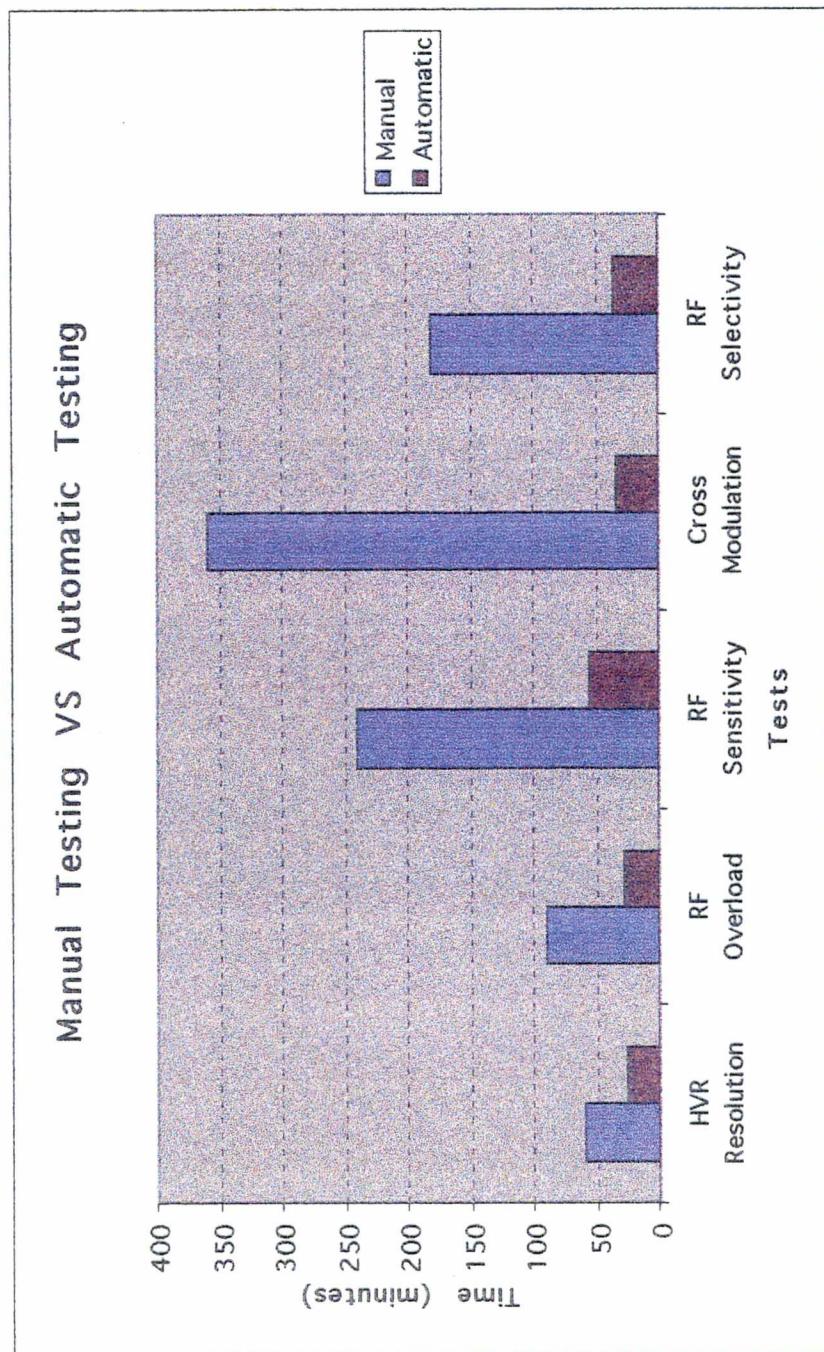


Figure 6.1 Manual Test Times Vs Automatic Test Times

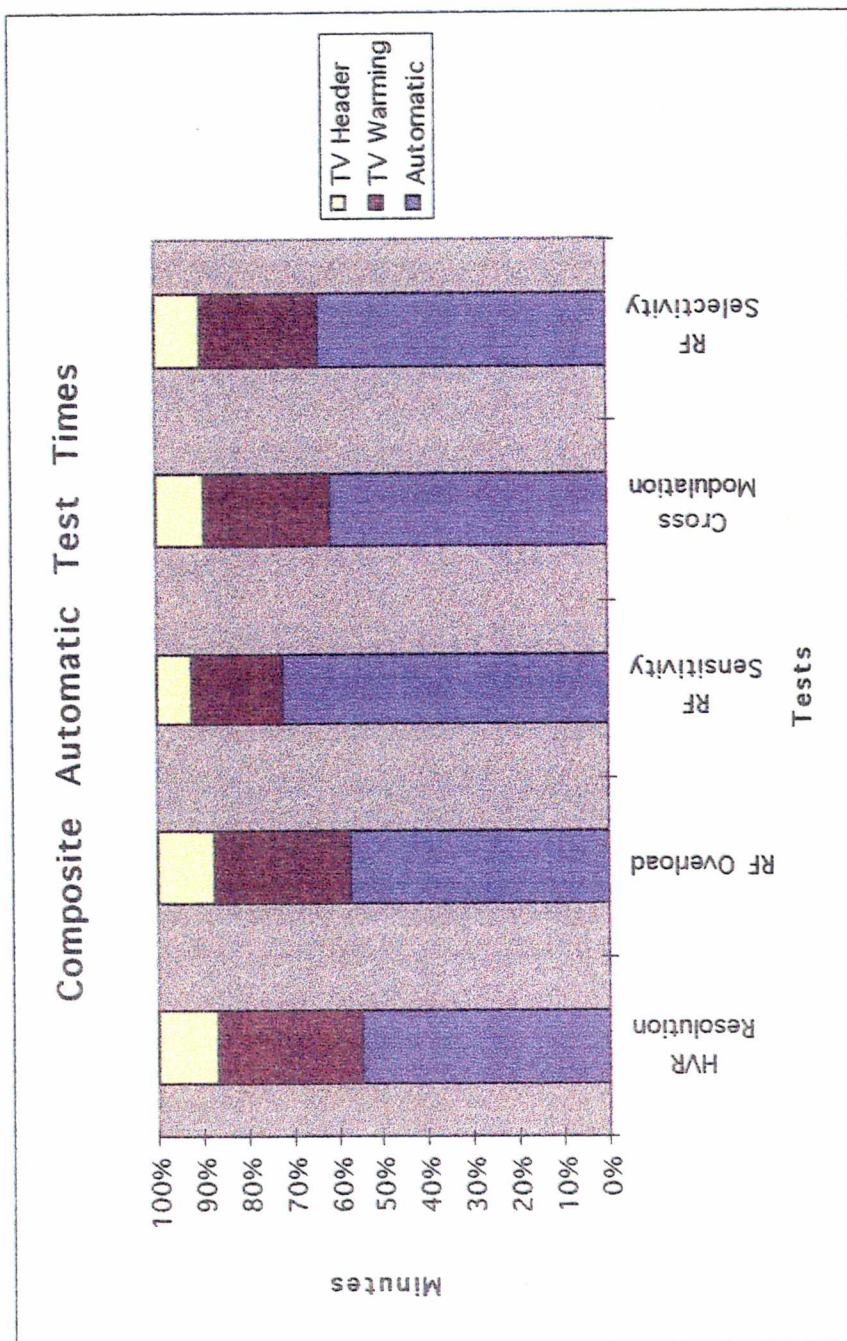


Figure 6.2 Composite of Automated Test Times

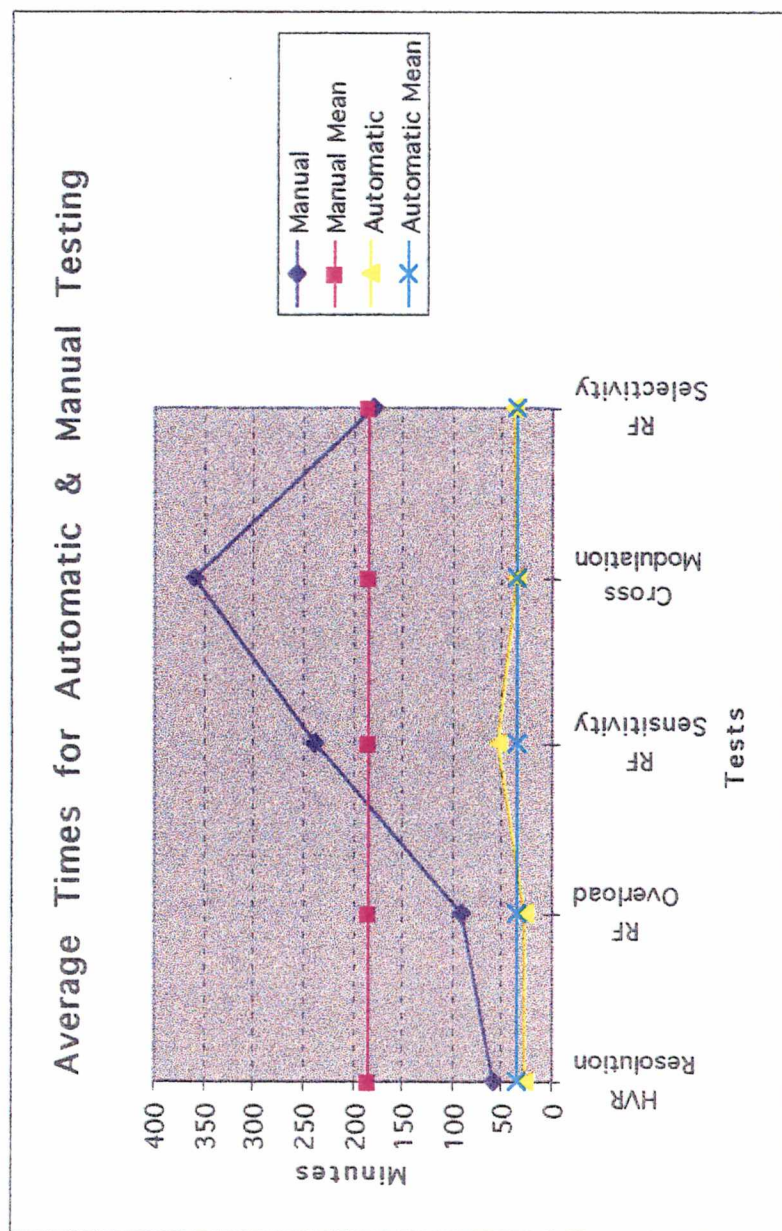


Figure 6.3 Average Test Times for Automatic & Manual Testing

minutes; on the other hand, the automated test results averaged just over one half hour, at 36 minutes. The results clearly show that automated testing for the tests executed, is considerably faster than manual testing for the same number of television. Figure 6.4 shows the productivity gained on a per test basis by charting the difference between the manual estimated times and the automated actual times.

This thesis encapsulates the initial stages of the ATF project. Many of the methodologies used to transform manual test procedures to useful and reliable automated test procedures were presented. The most prolific of these was the application of the Cleanroom Software Engineering process, although in partial form. The process provided a basis for the ATF Team to realistically aspire for zero-defect code on first execution. The rigor of the Cleanroom process requires more time to execute than more traditional methods; however, the pay-out in decreasing debugging time, as evident with the ATF project, is well worth the front end investment. Cleanroom's philosophy appears to be a recycle of an ancient quality cliché. Namely, if one does not have time to do a job correctly the first time, when will he have the time to do it again. The author agrees strongly.

Incremental Development by design sometimes results in overlapping increments. The need for two or more increments being developed simultaneously is often the case to adhere to a

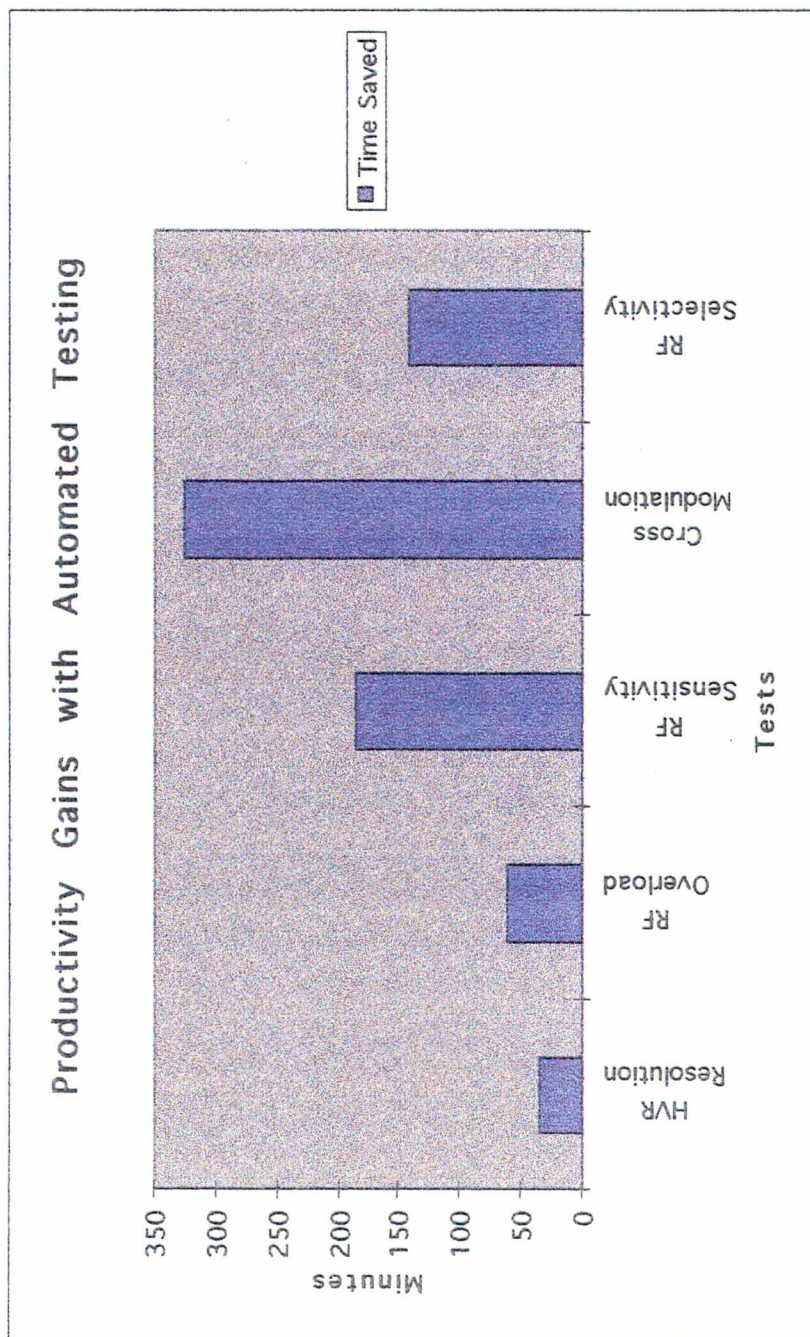


Figure 6.4 ATF Productivity Gains

schedule. The process should be monitored; otherwise, this could lead to see-saw changes- changing event A to event B and B back to event A.

Although the ATF team had good group dynamics, initiative, participation, and work ethic, there are a couple of areas that could be honed to improve efficiency. The team utilized some well known quality tools to derive many team conventions, assignments, and guidelines. These included brain storming and multi-voting to ensure team involvement and ownership. To ensure that these efforts are not minimized or lost, single point accountability could be assigned for various tasks. Monitoring and controlling key elements of the process become increasingly more important as the project's size increases.

Another improvement opportunity that is directly related to quality control is the development of a method for measuring errors for graphical languages. Viable measurements of graphical systems must be attainable in order for a process to be statistically controlled as prescribed by Cleanroom. Then, excellence levels for graphical program development could be quantified like their textual counterparts.

LIST OF REFERENCES

LIST OF REFERENCES

1. Colloms, Martin, *Computer Controlled Testing and Instrumentation*, John Wiley & Sons, New York, 1983, pp. 1-5.
2. Buckroyd, Allen, *Computer Integrated Testing*, John Wiley & Sons, New York, 1992, pp. 57-108.
3. Whitaker, Jerry, Benson, K., *Television Engineering Handbook*, McGraw-Hill Inc., New York, pp. 4.3-4.35, 12.3-12.21, 13.-13.147.
4. Ingram, Dave, *Video Electronics Technology*, Tab Books Inc., Blue Ridge Summit, PA, 1983, pp. 47-86.
5. Knowles, Roy, *Automated Testing*, McGraw-Hill, UK, 1976, pp. 1-40.
6. ATF Team Notes, *Introduction to Color Televisions*, Knoxville, TN, 1994, p.3.
7. Mazda, F., *Electronic Instruments and Measurement Techniques*, Cambridge University Press, New York, 1987, pp.58-59.
8. Coristi, Anthony, *IEEE- 488*, Academic Press Inc., San Diego, CA, 1989, pp. 51,52,102.
9. Sun Microsystems, *1.05 Gbyte Low Profile Disk Drive Manual*, Mountain View, CA, 1993.
10. Tektronix, Inc., *TekXpress Family of X Terminals Reference Manual*, Wilsonville, OR, 1983.
11. Tektronix, Inc., *TSI 8150 Test System Interface Reference Manual*, Wilsonville, OR, 1989.
12. Sierra Corp., *Instruction Manual*, Grass Valley, California.

13. RE Inc., RE 540 Operations Manual.
14. Rhode & Schwarz Inc., *Signal Generator SMT Operating Manual*, Germany.
15. Philips Inc., *PM5640A Operations Manual*, Indianapolis, Indiana.
16. Staco Energy Products Co., User's Handbook for MP series controller, Dayton, OH, 1993.
17. Tektronix VM700A, *Operations Manual*, Wilsonville, OR.
18. Tektronix Inc., *2714 Spectrum Analyzer Programmer Manual*, Beaverton, OR, 1994.
19. Magtrol Inc., *Magtrol 5100 Single Phase Analyzer Users Guide*, Buffalo, New York.
20. Scientific Atlanta, *Model 6350 CCIR-M (NTSC) Television Modulator*, Atlanta, GA, 1993.
21. Stover, Allen, *ATE: Automatic Test Equipment*, McGraw Hill Book Co., New York, 1984, pp.3-26.
22. Kodosky, Jeff, "Objects and Messages in the LabVIEW Graphical Programming System", 1993.
23. Kodosky, Jeff, "The LabVIEW Graphical Programming System", 1993.
24. Jagadeesh, J., Wang Y., The IEEE, Inc., "Reprint from Computer Magazine", 1993.
25. National Instruments, Inc., *LabVIEW Instrument 1/0 VI Reference Manual*, 1994.
26. Schmeisser, K., Barber, J., "Developing a LabVIEW 2 Instrument Driver", 1991.
27. Mills, Harlan, Poore, J.H., "Bringing Software Under Statistical

Quality Control”.

28. Linger, Richard C., “Cleanroom Process Model”, IEEE CS Press, Los Alamitos, CA, 1993.
29. Trammell, Carmen, Pleszkoch, Mark G., Linger, Richard, Hevner, Alan, “The Incremental Development Process in Cleanroom Software Engineering”.
30. Trammell, C.J., Linger, C.R., Hausler, A.P., “Adopting Cleanroom Software Engineering with a Phased Approach”.
31. Grant, Eugene L., Leavenworth, Richard, *Statistical Quality Control*, McGraw Hill Book Company, New York, 1988.
32. Trammell, Carmen, Personal Correspondence, “User Interface Specification”, University of Tennessee, June 1994.
33. Singh, Keeratpal, Personal Correspondence, CISP Group, University of Tennessee, June 1995.
34. Shah, Sameer, Personal Correspondence, CISP Group, University of Tennessee, June 1995.
35. Wan, Qin, Personal Correspondence, CISP Group, University of Tennessee, June 1995.
36. Ragsdale, Gary, Personal Correspondence, CISP Group, University of Tennessee, June 1995.

VITA

Eric Bernard Wrushen was born in Memphis, Tennessee on January 17, 1968. He graduated from Memphis Central High School in May of 1986. He attended The University of Tennessee at Martin (UTM) for two years where he majored in Electrical Engineering (EE). During his freshman year at UTM he participated in the Cooperative Engineering program (Co-op) by working alternate semesters with the local utility company in Memphis. Eric transferred to The University of Tennessee at Knoxville (UTK) in the Fall of 1988. He continued to Co-op for two more terms. Eric graduated from UTK in May, 1991. Upon graduation he went to work as a Systems Engineer and Team Leader for a national consumer products company. He returned to UTK to work on his Master's degree in EE a year and a half later. During his tenure at UTK Eric also worked as a research assistant on the ATF project. He graduated in August of 1995.