IS-4991
UC-37

AMES LABORATORY

Artemis User Guide

Version 2.1

M. S. Gorbics

Ames Laboratory
Iowa State University
Ames, Iowa 50011

Prepared For
The U. S. Department of Energy
Under Contract W-7405-eng -82

DISCLAIMER

Printed in the United States of America

Available from
National Technical Information Service
U.S. Department of Commerce
5265 Port Royal Road
Springfield, VA 22161

IS--4991

DE90 011719

Artemis User Guide
Version 2.1

M. S. Gorbics
Ames Laboratory* and Department of Physics
Iowa State University
Ames, IA 50011

Date Transmitted: February, 1989

*Operated by Iowa State University for
the U. S. Department of Energy under Contract
No. W-7405-ENG-82.

Table of Contents

Abstract

This document describes a software package used to test the Ames Waveform Digitizer. It is intended to help the user in setting up and operating a test bench using the FASTBUS (IEEE-960) data acquisition protocol connected to a VAX^{tm} computer.

1. Introduction

Artemis is the software part of a test bench used to evaluate the performance of the Ames Waveform Digitizer. Artemis allows the operator to exercise the digitizer in a variety of configurations. The testbench in Ames consists primarily of a FASTBUS crate including a host interface to a VAX and a General Purpose Master (GPM), a VAX/VMS system, and a Hewlett Packard 3314A function generator. The user directs Artemis to perform tests by selecting menu options and results are displayed as tables and graphs.

This document is intended to aid both the users and implementors of this program. In the next section we describe the structure of the program and its interfaces with the hardware. This will facilitate additions and modifications to Artemis and the testbench hardware. Because this program has been developed in a testing environment it has been designed for easy modification. Once the structure is understood, it should be possible to do many modifications without invalidating any of the testing programs. While the user interface makes the addition of new tests and equipment easy, the Artemis program is very specific to the VAX/VMS system.

The section on Tests and Services describes the tests which are currently available in Artemis. These tests include primitive scope loops, measurements of linearity and differential linearity, and general shakedown tests of the digital systems.

The user is presumed to be familiar with the Ames Waveform Digitizer Module User Guide.

Correspondence about this document should be addressed to W. T. Meyer, Room 12 Physics, Iowa State University, Ames, IA 50011.

2. Program Organization

Artemis consists of a group of relatively independent modules. Each module performs a particular set of operations on the FASTBUS system. Depending on the task intended, reasonable assumptions are made by the module. For example, the linearity testing module expects the function generator to be connected to the digitizer input, the FASTBUS to be initialized and the correct software already loaded to the digitizer's microprocessor before it begins. However, the FASTBUS primitives module will operate even without the digitizer in the FASTBUS crate and can be very useful for testing other FASTBUS modules.

The main routine in Artemis presents a list of the options. Each option selects one of the other modules. Most modules will, in turn, present their own menu of tasks which can be performed. When the operator closes the module, it will terminate and restore the terminal screen as it was when the module was initiated. In this way, Artemis runs continuously, starting and stopping tests as directed.

2.1 Program Modules

The Artemis program is stored in a root directory with the logical name art$root. The program is initiated with the command '$ run artemis/nodeb.' All of the files required by Artemis are accessed with logical names. These logical names are usually directed to a lower node in the directory tree.

The source code for the primary modules is stored in the directory art$root:[source]. Each source code file is a command file which creates a FORTRAN source file, menu text file, and any include files needed by the module. This same command file then compiles the source code and purges any excess files. All FORTRAN commons are implemented as include files in the directory art$root:[source.common]. The text for menus is located in the subdirectory art$root:[source.menutext]. These directories are referred to by the logical names 'common' and 'menutext.'

Finally the executable image is created, in the Ames testbench, with the command '$ @ames_link'.

2.2 Menus and Displays

Menus and displays are generated using the screen management routines (SMG) of the VAX/VMS system. This allows the Artemis menu system to be compatible with any terminal known by the VAX/VMS system. Each module is expected to respect the SMG system by restoring the contents of the terminal screen upon its termination.

The SMG system provides for the creation of 'virtual displays.' Each display is like a note card. The number of rows and columns in the display is determined when the display is created. Users can put any characters they want in the display and save it in an unseen place. Then, when desired, the note card can be posted in any desired position on

the terminal screen. When the card is removed, any cards that were covered will become visible again.

Typically, the first time a module is activated, it will open its corresponding file in the menutext directory and read all the text for its displays into memory. These displays fall into three groups. First, there are the menus. In a menu display, the second column contains the letter to type in order to select the option described to the right of the letter. Second, there are the data displays, for which all the labels, units and explanation text are in the text file and the numbers are filled in later as they are generated. Finally, some displays contain only numbers and have little or no text to store in the text file. An example would be the display of the contents of the microprocessor memory where the display window is filled edge to edge with hexadecimal numbers. This type of display is usually created and deleted as it is needed, rather than saved in the menutext file.

Table 2.1. Support Routines for Menus and Displays

MENU_PICK(ID)	Wait for menu selection
DISPLAY_WAIT	Wait for any key
PLOT_WAIT	Wait for any key
WARN(ID)	Display simple warning
NEW_DISPLAYS(FILENAME)	Open text file of display data
GET_DISPLAY(ID)	Read text of one display
END_DISPLAYS	Close display file
SHOW_X(ID,IR,IK,IV)	Hexadecimal number
EDIT_X(ID,IR,IK,IV)	Modify
SHOW_F(ID,IR,IK,AV,FMT,NCOL)	Floating point number
EDIT_F(ID,IR,IK,AV,FMT,NCOL)	Modify
SHOW_D(ID,IR,IK,IV,FMT,NCOL)	Decimal integer
EDIT_D(ID,IR,IK,IV,FMT,NCOL)	Modify
SHOW_C(ID,IR,IK,CV,FMT,NCOL)	Character string
EDIT_C(ID,IR,IK,CV,FMT,NCOL)	Modify
SHOW_A(ID,IR,IK,IV,OPTS,NOPT)	Show one of a set of strings
EDIT_A(ID,IR,IK,IV,OPTS,NOPT)	Select new option

Because these displays are common throughout Artemis many support routines, located in the module named SMG have been written for them. Table 2.1 lists the primary routines in this module along with a short description. The variable ID is a unique integer assigned by the system to each display. IR and IK indicate the row and column where the action is to take place. All row and column locations are relative to the upper left corner of the display, not the terminal screen. IV, AV, and CV are integer, floating point and character variables, respectively, and contain the current value of the variable being modified or displayed. FMT is a character string containing the desired FORTRAN format specifiers in parenthesis and NCOL is the number of characters of output the format will generate.

The MENU_PICK routine takes the display indicated and highlights the second column (assuming these are the allowed options) and waits for the operator to select an option.

The routine DISPLAY_WAIT is used by MENU_PICK and waits for the user to type a key. Most characters typed by the user are returned in the character variable 'IC' in common 'VDCOM', but a few characters are interpreted immediately. The 'Z' option (not listed in many menus, but always available) will cause the current text screen to be saved to a file for printing. The '!' option will cause a screen refresh.

The routine PLOT_WAIT is used for graphics displays. This routine insures that all graphics characters are sent to the screen and waits. When the user presses a key, the corresponding character is returned as described above. Again, the 'Z' option will save the contents of the screen for printing. The graphics commands to generate the screen are always saved in a file, but if the 'Z' option is not selected the file is deleted. Thus the operator cannot print the same screen twice, unless it is erased and redrawn. This system also allows access to any higher resolutions that may be available on the plotting device.

The routine WARN plants the indicated display in the middle of the screen and waits for the user to type a character (returned in IC). Generally this is used to warn the operator that something is not right.

The three routines named 'xxx_DISPLAYS' are used for retrieving text for displays from the 'menutext' directory. The first opens the file indicated. The second reads one display and stores a pointer to the display in the variable 'ID'. This pointer must be used anytime a reference to the display is made. Finally, the last routine closes the file.

Table 2.2 shows an example of the format for the menutext file. The first two numbers indicate the number of rows and the number of columns. Experience has shown that keeping the displays basically the same width is easier on the eyes, so the standard width is 32 columns for small menus and displays and 80 for full screen monsters. The next line in the menutext file is the title, which will be in the center of the upper edge of the display. The following lines will be the initial text in the display. In this example, dots indicate where numbers will show later.

The standard VAX routines in the SMG system are used for functions such as creating displays, putting them on the screen (paste), and removing them. Consult the VAX reference manual or the Artemis source code for details. The convention for positioning the menus on the terminal screen is to overlap them like pages, so that the headers of covered menus are still visible. This requires that each level of menus be lowered one additional line and indented two additional columns. See section three for examples of the terminal screen.

The group of routines 'SHOW_x' and 'EDIT_x' is designed to modify numbers and character strings in a display. The current contents of the display at the location given are overwritten with the new data. In the case of the edit, the current value of the variable is shown highlighted and the operator may accept the number as it is or enter a new value.

Table 2.2. Example text for display routines

```
  4 32
Menu Title
 T Tic Tac Toe
 C Chess
 G Global Thermonuclear War
 E Exit
  4 32
Display Title
Temperature    .... F
Pressure       .... KPa
O2 content     .... ppm
Radiation      .... rem/hr
```

These routines are fairly tolerant of entry errors. If the error is intercepted, the operator is prompted to reenter the data.

2.3 Plots, Graphs and Fits

Plots and graphs are used throughout Artemis. The module PLOT provides routines for drawing histograms and point plots. Additional options allow the user to select one of several types of fits to be superimposed on the data. The same fits are available for analysis of the data within the Artemis modules. Plot limits, plot type, fit type, and titles are maintained in a data base which may be modified by the operator.

Table 2.3 lists the basic entry points for the plotting and fitting routines. The variables X and Y are arrays of NPTS points each. The variables BQ, BL, BIL, B, AL, AQ, A1, A2, B1, B2 are fit parameters in various formats. At present no uniform naming convention for these variables exists. Consult the source code for the current usage of these routines.

Table 2.3. Plotting and Fitting Routines

```
QPLOT(NPTS,X,Y,BOUNDS,IATTR,LABEL)
QPLOT_EDIT(BOUNDS,IATTR,LABEL,IPLOTS)
FUNCTION CHISQQ(X,Y,NPTS,BQ)
FUNCTION CHISQL(X,Y,NPTS,BL)
FUNCTION CHISQBL(X,Y,NPTS,BIL)
QFIT(X, Y, NPTS, BL, BQ)
TWOFIT(X, Y, NPTS, BREAK, B)
FOURFIT(X, Y, NPTS, IP, AL, AQ)
BIFIT(V, A,NPTS, BREAK, A1, B1, A2, B2, CHISQ)
```

The routine QPLOT generates a graphic display of the data given. Every plot is marked with the currently selected channel and the saved timestamp string (presumably

corresponding to when the data were collected). If a fit has been specified, it is done only to the plotted data points and then displayed. Some parameters of the fit are also displayed to the left of the plot. No provision is made to save the text on the terminal screen. This must be done before a call to QPLOT. QPLOT returns as soon as all the graphic output is generated. The calling routine may then add additional graphics to the screen before calling PLOT_WAIT. After PLOT_WAIT returns, the calling routine must either begin a new plot or restore the previous menu data to the screen (see SMG routine SAVE_PHYSICAL_SCREEN).

The graphics support is obtained with a modified version of Mini GD3 on the VAX system. The modifications are all associated with the implementation of the 'Z' option to print graphic displays.

The variables BOUNDS, IATTR, LABEL, and IPLOT form the data base of plotting parameters. Included are the plot limits, attributes and labels for all the plots in a single Artemis test. Table 2.4 shows the format for these variables. The initial values of the plotting data base are declared in each Artemis module. This data base can be modified either by the Artemis module or a call to QPLOT_EDIT. The routine QPLOT_EDIT puts up a data display of the current values and allows the operator to modify any of the values.

Table 2.4. Plot database variables

BOUNDS	(1,I)	Lower x limit of I-th plot
	(2,I)	Upper x limit
	(3,I)	Lower y limit
	(4,I)	Upper y limit
	(5,I)	X grain size (not used)
	(6,I)	Y grain size (not used)
ATTR	(1,I)	Number of tick marks, x
	(2,I)	Number of tick marks, y
	(3,I)	Fit*100 + Autoscale
	(4,I)	Plot type
LABEL	(1,I)	Text label
IPLOT	(1)	Current plot to edit
	(2)	Total number of plots

Note that space has been allocated for autoscaling (grain size) but this has not been implemented. This has not been a problem since many routines adjust the x-scale automatically. If it is implemented, it should be turned off and on by the parameter in ATTR(3,I).

The different types of fits are simply numbered. If the fit number is nonzero then the corresponding fit is done to the data between the upper and lower x-limits of the plot. Not all fits will work on arbitrary data. Some require the data to cover some specific ranges in either x or y. These fits are specialized to the FADC used on the digitizer.

The basic fitting routines are listed in Table 2.3. The routine QFIT performs both a linear and a quadratic fit to the data given. The variables BL and BQ are one dimensional vectors of the fitted linear and quadratic coefficients respectively. The constant term is first and the quadratic term is last. The routines TWOFIT and FOURFIT partition the data according the multi-linear nature of the the FADC and perform fits (with QFIT) on each partition separately. BIFIT is similar to TWOFIT except that the two partitions are constrained to meet at the point defined by BREAK.

2.4 FASTBUS interface

Where possible Artemis uses subroutine calls which conform to the NIM FASTBUS standard. Where the use of the LRS 1821 host interface made it necessary, special FASTBUS interface routines were developed to conform to this standard. These routines are stored in a file called FB21.FOR. With the introduction of a similar set of FASTBUS routines and a few modifications Artemis should be able to use a different FASTBUS interface. In many set-ups, Artemis should be able to use only the standard routines. An earlier version of Artemis has been successfully run at CERN using the standard CERN libraries and CFI to VAX interface.

2.4.1 Software

Artemis uses only a subset of the routines and features defined in the standard mentioned above. In this section, we describe the interface routines as implemented for use with Artemis and the LeCroy 1821 FASTBUS master.

We use the arrays STATUS and CNTRL throughout the FASTBUS interface routines. Their functions are as their names suggest. While the standard describes complex formats for these arrays, our routines use only the second word in each of them for controlling and reporting the results of a single operation. The standard also supplies two names for each function, but we only use the short version. The names are made from code letters which indicate Read, Write, Control space, Data space, Block transfer, and Multi-listener (Broadcast operation). Table 2.5 is a list of subroutines defined in FB21.FOR.

The PRIMAD and SECAD variables indicate the primary and secondary address, respectively. OUTBUF and INBUF are single word or array variables for the data being handled. MAXLEN is the number of bytes for a block transfer. This version requires that this number be an exact multiple of four to allow translation into FASTBUS cycles (division by four).

The routine FWDBX is non-standard. This routine performs a block transfer write operation, but only a single data word is accepted as input. This data word is written multiple times to the target slave. In Artemis, this routine is used to zero large blocks of the microprocessor program memory before the initial program load. This routine was made necessary because of a bottleneck in the VAX to 1821 interface in Ames.

Table 2.5. FASTBUS Interface Routines

```
FBINIT (STATUS, CNTRL)

FRD (STATUS, CNTRL, PRIMAD, SECAD,  INBUF)
FWD (STATUS, CNTRL, PRIMAD, SECAD, OUTBUF)
FRC (STATUS, CNTRL, PRIMAD, SECAD,  INBUF)
FWC (STATUS, CNTRL, PRIMAD, SECAD, OUTBUF)

FRDM (STATUS, CNTRL, PRIMAD, SECAD,  INBUF)
FWDM (STATUS, CNTRL, PRIMAD, SECAD, OUTBUF)
FRCM (STATUS, CNTRL, PRIMAD, SECAD,  INBUF)
FWCM (STATUS, CNTRL, PRIMAD, SECAD, OUTBUF)

FRDB (STATUS, CNTRL, PRIMAD, SECAD, INBUF, MAXLEN)
FWDB (STATUS, CNTRL, PRIMAD, SECAD,OUTBUF, MAXLEN)
FWDBX(STATUS, CNTRL, PRIMAD, SECAD,OUTBUF, MAXLEN)
```

Table 2.6 lists the function of the bits in the second word of the CNTRL array. For compatibility, the first word of the control array should contain the value sixteen (indicating a length of the array in bytes), but this is not checked in this version.

Table 2.6. Meaning of bits in CNTRL(2)

0	FOGERR	Leave bus state unchanged on ANY error abort
1	FONOAR	No Arbitration Cycle
2	FONOPA	No Primary Address Cycle
3	FOEG	Master asserts EG (Geographic address)
4	FOGKUP	Retain Bus Mastership at end of operation
5	FONOSA	No Secondary Address Cycle (Internal Address)
6	FONDC	No Data Cycle(s)
7	FOASUP	Retain AS/AK lock at end of operation
8	FOBLKE	Define SS=2 as Block Transfer error end
9	FOBUFE	Define MAXLEN transfers as Block Transfer error end

Table 2.7 describes the meaning of the status bits in the second word of the STATUS array. The first word of the status array should also contain the value of sixteen to indicate the length of the array. This value is not checked in this version nor are any words, other than the second, used or modified. A routine, FBCHECK, exists to allow the user to check STATUS(2) and write an appropriate error message if it is non-zero.

2.4.2 LeCroy 1821 Software

The material in this section is specific to the LeCroy 1821 Segment Manager Interface, which we use in Ames. Users of other systems may skip this section.

The software for the LeCroy 1821 FASTBUS interface is organized in five parts. The FB driver is installed on the microVAX and provides the interpretation of QIO calls

Table 2.7. Meaning of bits in STATUS(2)

0	FESS0	SS0 line value
1	FESS1	SS1 line value
2	FESS2	SS2 line value
3	FEARB	Failed to obtain Bus Mastership
4	FETMAK	Timeout on AK handshake
5	FEAKSS	Bad SS response with AK
6	FEREL	Failed to release Bus Mastership
7	FETMDK	Timeout on DK handshake
8	FEDKSS	Bad SS response with DK -or-
	FEPTRY	Read Data parity error (SS=0)
9	FEASAK	AS=1 or AK=1 before an address Cycle
10	FEASRL	AS=1 or AK=1 after AS/AK disconnect
11	FEDSDK	DS≠DK before a data cycle
12	FEWTAC	Timeout on WT in a Address Cycle
13	FEWTDC	Timeout on WT in a Data Cycle
14	FEBLTR	Block Transfer End error (user defined)
15	FEINT	Inclusive OR of bits 3 and 4 or nonFASTBUS hardware error

directed to the FASTBUS "device." The FASTBUS device is named FBAn:, where n (normally zero) is selected by a switch on the front panel of the 1821. The file named PORTS.COM contains the routines (QIO calls) for the most basic transactions with the 1821. These transfer data to and from the eight 16-bit registers of the 1821. The file ROM_ISU.LRS contains the code necessary for the 1821 itself. The file U1821.COM contains general purpose routines for transferring data to and from the 1821. These routines will load and unload entire programs into the 1821 primary program or auxiliary memory. Finally the file FB21.FOR contains the routines to implement FASTBUS operations as described in the previous section.

The 1821 contains a high speed programmable sequencer (computer) specifically designed to interact with the backplane of the FASTBUS. The state of FASTBUS signals can be examined and changed every 5 ns. The connections to the VAX are done through the AUX connector of the FASTBUS crate (higher number of the two possible slots). A small card provides the connection to two 40 conductor ribbon cables which connect to a DR11W. The DR11W is a standard unibus device but the microVAX requires an adapter to connect the Q-bus to unibus.

The host computer controls the LeCroy 1821 either by downloading software or by selecting software from one of several ROMs in the 1821 (only the download of ROM_ISU.LRS is used in Artemis). The host then starts the 1821 in its idle loop. To execute an operation on the FASTBUS, the host computer sends the transfer address of a subroutine located in the program memory of the 1821. As soon as the 1821 returns to the idle loop, after completing the previous operation, it executes a jump to the address sent.

If the host computer were fast enough (or the 1821 slow enough), operations could be lost due to a pileup effect. Hardware flags are provided to prevent this problem by allowing the host to synchronize with the 1821. In practice, with conventional FASTBUS operations, the microVAX is too slow for this to be a problem except possibly on block transfers. Observed speeds for block transfers into the 1821 from the FASTBUS are more than 4 MHz. Thus a kiloword transfer would complete in less than 300 microseconds.

Data transfer to the host is done by manipulating a pointer into the data memory of the 1821. The host computer sets the address of the data memory pointer and reads the data. When the 1821 is reading data from the host, this same pointer is used and incremented as necessary.

The program memory of the 1821 is organized as 256 words containing 96 bits each. Every instruction uses exactly one word in the program memory. The instruction word is divided into various fields which control different subsystems of the 1821. For example one field in the instruction word will determine the state of the FASTBUS control lines (MS,SS,EG,RD, etc.) at the end of the instruction while another will independently determine if data are received from the bus or asserted on the bus. Incorrect FASTBUS protocols can be easily realized. For example, it is possible to assert data on the bus while directing a slave device also to assert data on the bus. Appendix 1 is a summary of the fields in the 1821 instruction word.

To help create new LRS-1821 software one major software tool was developed. This is a cross dis-assembler which converts the machine language of the LRS-1821 into mnemonics. While the LRS dis-assembler still requires the programmer to write in machine language, it provides a rapid method of checking the code created. The disassembler operates on a text file which lists each instruction and its location in hexadecimal. The output has the exact same format except that mnemonics have been added after each instruction word. Thus the programmer can make a change with a text editor, put the result through the dis-assembler, check the results, and either repeat the process or send the file to the loader (which ignores the comments and mnemonics). An example of this format is shown in appendix 2. Appendix 2 is the 1821 code developed for the test bench and supports all forms of addressing in FASTBUS (ROM_ISU.LRS).

The 1821 code is arranged into three basic sections. The first is the idle loop in locations 0-8 of the program memory. This loop allows the 1821 to watch for various conditions while waiting for the next instructions. For example, service requests could be monitored between FASTBUS operations. The second section, 9 through 3F(hex), is a jump table. Each subroutine starts in this section and if more that one or two instructions are required a jump is executed into the third section to reach a longer subroutine. This is useful because many subroutines are the same except for the setting of some status bits. For example both the geographic and logical primary address cycles are identical except for some status bits which can be set by the single jump instruction. This arrangement also allows the programmer to arrange the subroutines in the third section as necessary

without changing the transfer addresses stored in the host computer.

2.4.3 Common Interface Problems

On the Ames test bench some common errors with the FASTBUS interface can cause strange and confusing symptoms. In this section, we will describe some of the understood problems and their solutions. We hope information in this section will help prevent some head-scratching sessions in the lab.

Suppose the operator arrives in the morning and, after turning on the system, the power is OK but FASTBUS won't respond. The first thing to check is the status of the FASTBUS driver with the VAX command '$ show device fba'. If the device is offline this is the problem. Sometime during the night something caused the microVAX to reboot. When the FASTBUS driver was installed it could not detect the 1821 and went offline. There is no way to put the driver online. There are two possible solutions to this situation. Simplest is to reboot the microVAX with the FASTBUS power on. Second is to create another FASTBUS driver unit number. In other words, change the unit number on the 1821 and install FBA1: as a new device. This requires an expert.

Another situation is that everything is running smoothly and suddenly all transactions with the FASTBUS fail. This may be caused by a problem with the DR11W in the unibus interface. The easy solution is to turn the power to the DR11W off and then back on. Take care to not do this while access to the FASTBUS is being attempted by the VAX. This will cause a hard error and reboot by the VAX.

2.5 GPIB interface

The General Purpose Interface Bus (GPIB) is an IEEE standard now available on many electronic devices on the market. In the Test Bench we use this bus to interface to a HP 3314A function generator. This device provides the waveforms for most of the analog tests done on the digitizer boards. The interface board attaches directly to the Q-bus of the microVAX.

Software supplied with this interface allows a privileged user on the VAX to perform any operation defined on the GPIB, using the QIO calls of the VAX. Simple software modifications were required to allow a user access to the GPIB port without having exotic privileges from the operating system.

Typically a software package is written for using each device in the desired mode. For example the HP function generator is used as a millivolt source to drive DC levels to the input of the digitizer. A single subroutine call makes the connection to the function generator and selects the desired functions. Additional calls select only new DC levels to avoid the overhead of resetting all the possible functions of the generator. Other routines use the generator as a pulse generator and as a source of a sawtooth signal.

3. Tests and Services

The purpose of Artemis is to test Ames Waveform Digitizers. This section describes most of the basic services provided. Note that these tests are divided into three groups by their level of complexity. The lowest level is the FASTBUS primitives section which can operate even without the digitizer. The next level is other 'primitives' which do not require the digitizer's microprocessor to operate. At the highest level are the 'tests' which require the microprocessor to operate and, in general, use it to process large amounts of test data.

Access to these tests is through the menu system described earlier. Table 3.1 shows the appearance of the terminal screen when Artemis begins. The top line of the display is a time stamp from the last time data were read from a board and the serial number of the board used to collect the data. The initialization routines may modify these values.

Table 3.1. Main Menu

```
+--- 21-NOV-88 13.37.10 #693 ----+
| A Re-initialize Stuff          |
| B Fastbus Primitives           |
| D Digitizer Primitives         |
| M 68000 Primitives             |
| T Threshold set                |
| E Linearity Test               |
| F Board Tests                  |
| P Pulse Test                   |
| S SawTooth Test                |
| G Delta test                   |
| H Multi event scan             |
| W Bookkeeping                  |
| Z Snapshot (any menu)          |
| Q Quit (any menu)              |
+--------------------------------+
```

Note that the Z and Q options are displayed in this menu. These options are available in all menus but often are not displayed. The output of the snapshot will be put in the directory art$root:[output]. This output can be redirected to a printer with the logical name LASER.

3.1 Re-initialization

Selecting the Re-initialization option will result in the screen shown in table 3.2. The most frequently used option in this menu is the full initialization. The routine assumes the power to the FASTBUS crate has just been turned on. The host interface and GPM

are initialized and their programs reloaded. The digitizer board is initialized and its microprocessor programs are reloaded. If all is working the operator should be able to go directly to any of the tests at this point.

Table 3.2. Re-initialization services

```
+--- 21-NOV-88 13.37.10 #693 ----+
| +------- Initialization Services --------+
| | A Full initialization after power up   |
| | B Init Host Interface only             |
| | C Search Crate for Digitizer boards    |
| | D Init all boards                      |
| | E Select primary board                 |
| | F Read and display 1821 Code           |
| | G Init GPM only                        |
| | H SPECIAL                              |
| +--------------------------------+-------+
```

The other options are used less often. Options B through E are subsets of A. Options F and H are normally used to identify DR11W problems by reading parts of the 1821 program from the 1821. If these test work the problem is on the FASTBUS itself.

3.2 FASTBUS Primitives

The primitive FASTBUS operations allow the user to execute most "normal" FASTBUS operations. Selection of this option in the main menu will produce the screen display shown in table 3.3.

The primary and secondary address are put in the data displays by the user and the results of the operation are displayed below them. All forms of addressing are available. Logical addressing can be done both with secondary address cycles (Logical I) or with an internal address (Logical II). Note that the internal address format is only valid in data space. Broadcast operations are done by specifying the broadcast address as the primary address and selecting the Broadcast addressing mode. The broadcast address is complex and generally a FASTBUS reference manual is required to construct the correct address. Some of the more complex operations, for example pattern attach, are not possible with this system. ALL values are displayed in hexadecimal.

Scope loops and other testing programs can be run in the GPM using software downloaded into the GPM program memory. Option G presents the user with a menu of the currently available choices. The currently set primary address is used to select the target board.

Scope loops can also be executed in the Lecroy 1821, if it is available. Usually, it is preferable to run the loop in the GPM, but special circumstances may require the 1821. Option L is used to select these scope loops. The code for the 1821 is found in the

Table 3.3. FASTBUS Primitives

```
+--- 21-NOV-88 13.37.10 #693 ----+
| +------- FASTBUS PRIMITIVES ---+-----+ +---- FASTBUS ADDRESSING ----+
| | A Select Addressing Mode          | |Mode           Geographic    |
| | B Select Control/Data Space       | |Space            Control     |
| | C Select Primary/Geographic Addr  | |Primary Addr       00000011  |
| | D Select Secondary Addr/Data      | |Geographic Addr    00000011  |
| | R Read                            | |Secondary Addr     00000000  |
| | W Write                           | |Data               00000000  |
| | G GPM Scope Loops                 | |Status             00000000  |
| | L 1821 Scope Loops                | +-----------------------------+
| | P Primitive 1821 Function         |
| | O Load data to 1821               |
| | I Unload data from 1821           |
| +-----------------------------+-----+
| Z Snapshot (any menu)          |
| Q Quit (any menu)              |
+--------------------------------+
```

subdirectory art$root:[LRS]. The convention for scope loops is that the code overwrites the region 40 to 90 hex in the 1821 program memory and the initial transfer address is 40 hex. The format for code loaded into the 1821 has been discussed in a previous section.

Because the 1821 scope loops operate at a very primitive level, the primary address of the target module is hard coded in the 1821 code. An interlock is provided such that if the digitizer is not detected in the correct slot (19 decimal), a warning is displayed. Typing a tab character at the warning will bypass this check.

At a deeper level, the operator may specify an address in the Lecroy 1821 FASTBUS master, where execution is to begin (option P). This allows the users to execute parts of addressing sequences or control individual FASTBUS signals. Data can also be loaded and unloaded from the 1821 with options I and O. These options are primarily used for debugging 1821 code and require an expert.

3.3 Digitizer Primitives

The digitizer primitives module uses the displays shown in table 3.4. This module allows the user to examine the results of single triggers in three different formats.

A typical sequence for the examination of a single trigger might be as follows. First select option A to set the desired channel, event, threshold and width. Threshold and width are the zero suppression parameters. The board channel number is the physical location of the channel on the board. Then the function generator is set with option G. This assumes the function generator is active and used as input to the digitizer. Next, option C (or both T and R) is selected. This causes a trigger to be sent to the board and

Table 3.4. Digitizer Primitives

```
+--- 21-NOV-88 13.37.10 #693 ----+
| +---- DIGITIZER PRIMITIVES ----+    +---- DIGITIZER ADDRESSING ----+
| | A Set Parameters             |    |Channel                   0   |
| | C Trigger and Read           |    |Event                     0   |
| | D Display Previous Readout   |    |Threshold                 0   |
| | E Display Current Data       |    |Width                     0   |
| | F Plot Current Data          |    |Board Channel No.         0   |
| | H Edit Plot parameters       |    |Target ADC Code          64   |
| | K Set all pedestals          |    |Upper Search Limit     2150 mV|
| | G Edit Function Generator    |    +-------------------------------+
| | T Trigger                    |
| | R Read (no trigger)          |
| | S Pedestal Summary           |
| | L Search for ADC code        |
| +------------------------------+
+--------------------------------+
```

a readout of part of the FEB. The data selected with option A are saved and available for display. Option D shows the data exactly as read from the digitizer, in hexadecimal. Option E shows the data in decimal, corrected for the effects of zero suppression. Finally, option F display the data as a plot.

Notice that this program module can also be used to set the width and threshold for single channels. The option K has been included to allow the operator to set the width and threshold for all channels to the same value. There is also another module which will set the width and threshold based on the pedestal of each channel.

As stated earlier, this entire section operates without the microprocessor. Thus the analog functions and zero suppression functions can be examined even if the microprocessor is not functioning. These services are also useful for other tests, to adjust the system before starting a long or complex analog test.

Notice the option L, which will search for a DC offset corresponding to a particular ADC code. This is useful in the preparation of other tests where the saturation or breakpoint is needed in volts. The search is done with a primitive method called binary division. The algorithm starts with the voltage range -100 mV to the 'Upper Search Limit'. The center value of the range is tested and the range is cut in half. This procedure is repeated nine times. This method will always terminate. Do not attempt to search for the ADC codes 255 or zero as this will not produce the desired result.

3.4 Motorola 68000 Primitives

Microprocessor primitives were developed to test the microprocessor and its programs. These functions allow the user to test microprocessor memory, load a program, set param-

eters, start program execution, initiate interrupts and examine the results of program execution. It must be noted that user interactions with the microprocessor are at a very primitive level. It would be prudent to develop complex software on a different 68000 system before attempting to use it in this system. Table 3.5 shows the terminal display for this module.

Table 3.5. M68000 Primitives

```
+--- 21-NOV-88 13.37.10 #693 ----+
| +-------- 68000 PRIMITIVES ----+----+ +-------- 68K STATUS --------+
| | A Zero 68K Memory                | |M68000 State     HALTED      |
| | B Test 68K Memory                | |Program Name     Artemis     |
| | C Display/Edit Comm Vector       | |Last Interrupt           6   |
| | L Load 68K Program               | |Secondary Addr     0004000   |
| | D Display 68K Memory             | +-----------------------------+
| | 1 Reset 68K                      |
| | 2 Halt  68K                      |
| | 5 Interrupt 5                    |
| | 6 Interrupt 6                    |
| | 7 Interrupt 7                    |
| | S Scope Loop                     |
| +----------------------------------+----+
| Q Quit (any menu)                |
+----------------------------------+
```

Option A quickly zeroes the memory of the microprocessor. This is usually necessary only for clarity when the operator will be scanning the memory to see the effects of a program.

Option B is a memory test which does not use the microprocessor. A small display is created with 128 characters. As each kilobyte of memory is checked a 'P' or 'F' is displayed for 'pass' or 'fail'. The test is slow and any keystroke will terminate the test early. Faster and more thorough memory tests using the microprocessor are available in other modules.

The established convention for programs in this embedded processor is to use a communications vector begining at FASTBUS dataspace location 40 hex. Each word in the vector has a general name for the type of information it is used for. Option C will read this vector from the board and display it on the terminal. Typing a space will allow the user to edit the numbers in the vector. If the edit option is used, the vector will be written back to the same place.

Option L will load a program into the microprocessor memory. A file containing S-code will be interpreted and loaded with FASTBUS operations into the microprocessor memory. The default filename for S-code executables is art$root:[m68k].sex. This system may also be used to load standard event data into the FEB.

Option D will display sections of memory a screenfull at a time. A space will cause the next screenfull to be displayed. Any other key will terminate the display.

Option 2 will halt the microprocessor and hold it in the reset state. This is the initial state of the processor. Option 1 will release the processor from this state and it will begin executing at the reset vector. If the processor is not halted, option 1 has no effect.

Options 5, 6, and 7 will cause interrupts at those levels. The convention in programs is that interrupt 5 is used for initialization and interrupt 6 is used for execution. Since interrupt 7 occurs automatically at the end of a zero suppression cycle, several tests cause the program to transfer control to the code for interrupt 6. This method gives faster data collection.

The scope loop option causes the microprocessor to be reset and then initiates the test program in the 1821. This is, of course, hardware dependent. The loop is effected in the host VAX computer. This option is not needed when loops in the GPM are available.

The status display shows the state of the microprocessor according to the options selected in this program module. The display is updated only by routines in this module and not by examination of the microprocessor. Most of the processor control routines are available for use by other modules. These routines will keep the status display up to date.

3.5 Board Tests

Board tests primarily consist of a series of shakedown tests, each of which consists of a sequence of short tests. The shakedown test attempts to exercise quickly every known function of the board. Table 3.6 shows the 'board tests' menu.

Table 3.6. Board Tests

```
+--- 23-NOV-88 10.59.45 #693 ----+
| +------- BOARD TESTS --------+ |
| | C Coupler Shakedown        | |
| | F FEB Shakedown            | |
| | M M68K Shakedown           | |
| | P Pedestal Summary         | |
| +----------------------------+ |
```

3.5.1 Coupler Shakedown

The coupler shakedown will put up the display shown in table 3.7. As the test proceeds either a 'Pass' or a hex number will be displayed. A hexadecimal number indicates a failure. Bits which are set in this number show which bits were incorrect at the end of the test. The three columns show the same test with different addressing modes. The broadcasts are of the class N type, and at present this test will work correctly only with a single digitizer board in the crate.

Table 3.7. Coupler Shakedown Display

```
+--- 23-NOV-88 10.59.45 #693 ----+
| +------- BOARD TESTS --------+ |
| | +---------------------- COUPLER SHAKEDOWN ----------------------+
| | |Addressing Mode                    GEO       LOG       BRO     |
| | |Read  CSR#0 (Mfg. Number)          Pass      Pass      Pass    |
| | |Read  CSR#0 (Data Space)           Pass      Pass      Pass    |
| +-+Read  CSR#1 (Serial Number)        Pass      Pass      Pass    |
| F |Read  CSR#1 (Data Space)           Pass      Pass      Pass    |
| R |R/W CSR#11 (Trigger Acc. Number)   Pass      Pass      Pass    |
| P |R/W CSR#11 (Data Space)            Pass      Pass      Pass    |
| S |R/W CSR#3  (Logical Address)       Pass       NA       Pass    |
| G |R/W CSR#7  (Broadcast Class)       Pass      Pass       NA     |
| H |S/C Error flag                     Pass                        |
| W |S/C Logical addressing             Pass                        |
| Z |S/C LED                            Pass                        |
| Q |S/C Zero Suppress Override         Pass                        |
+---+Reset (clear error flag)          Pass                        |
    |Bus Reset (Clr Error Flag)         Pass                        |
    |General Broadcast to CSR#11        Pass                        |
    |Sparse Data Scan (DTP)             Pass                        |
    |Device available Scan (FFU)        Pass                        |
    |Device Present Scan                Pass                        |
    |Pattern Select                     Pass                        |
    +---------------------------------------------------------------+
```

The coupler shakedown test attempts all forms of addressing and exercises most of the bits that can be set and cleared. These tests include T-pin scans and pattern select.

3.5.2 Microprocessor Shakedown

The display for the microprocessor shakedown is shown in table 3.8. The test begins with a short memory test to check the location where a test program is then loaded. The 'interrupts' test will check that all three interrupts function and that an interrupt 7 occurs as a result of a trigger. The clock frequency is estimated by having the processor execute a loop which is timed by the host VAX. This test will detect if any wait states occur during instruction fetches by showing a low effective clock frequency.

The 'correct sieve execution' refers to a famous benchmark program which computes all the prime numbers less than about sixteen thousand. This test executes this benchmark a few times and the host computer checks that the microprocessor found the correct number of primes. This test should detect any execution errors in the processor.

In the 'coupler access' test the microprocessor checks that it can access parts of the coupler.

Table 3.8. Microprocessor Shakedown

```
+--- 23-NOV-88 10.59.45 #693 ----+
| +------- BOARD TESTS --------+ |
| | +--------- M68K SHAKEDOWN -+-+-----+
| | |Memory test (4Kb)       Pass      |
| | |Load test program       Pass      |
| | |Interrupts              Pass      |
| +-+Est. Clock Frequency     16.0824   |
| F |Correct Sieve Execution Pass      |
| R |68K Coupler Access      Pass      |
| P |Memory tests            Pass      |
| S |FB Coupler Access       Pass      |
| G +----------------------------+-----+
```

The 'memory tests' does a 'running bit' test to all of the memory except that holding the testing program, including the FEB and threshold memory. The bit pattern will indicate which of these three failed and examination of the contents of memory shows where the test was terminated.

'FB Coupler Access' checks that the microprocessor can operate correctly with interference from the FASTBUS. The test program is just a loop counting in a particular location. The host VAX program then accesses this location 100 times. Each FASTBUS access stops and starts the microprocessor by the required bus arbitration. If the microprocessor fails to restart, an error will be indicated.

At the end of the entire shakedown test, the default program is reloaded. Because this loading process is not particularly fast, there can be a somewhat mysterious delay in the response of Artemis to new commands.

3.5.3 FEB Shakedown

The FEB shakedown tests several features of the FEB. Table 3.9 shows the display during an FEB shakedown. The tests shown are not executed in the order shown. The write to the FEB is last as it takes a long time. Any keystroke will terminate this part of the test.

The first test executed is the FEB Event crosstalk. This tests the bits that select one of four possible event areas in the FEB to insure that in writing a given event we do not inadvertently overwrite one of the other events. The next test, FEB Channel crosstalk, is in the same vein. It tests to see if writing to one channel causes trash to be written into other channels. If a hex number is displayed as the result of the test, the bits set in the word indicate which channels failed.

The 'word count' test runs twice. First the zero suppression thresholds are set to zero, a trigger is given, and the word count for each channel is checked. Since no zero suppression will occur, all word counts should be 256. Note that this number is represented

Table 3.9. FEB Shakedown

```
+--- 25-NOV-88 10.03.16 #669 ----+
| +------- BOARD TESTS --------+ |
| | +------------------------ FEB SHAKEDOWN -------------------------+
| | |Write to FEB             Pass      Pass      Pass      Pass     |
| | |FEB Event Xtalk          Pass                                   |
| | |FEB Channel Xtalk        Pass                                   |
| +-+Word Count               Pass      Pass                         |
| F |Dump Count               Pass                                   |
| R |Valid Data Bit           Pass                                   |
| P |Zero Suppress Override   Pass                                   |
| S +---------------------------+-----------------------------------+
```

as a zero byte in the FEB. The second trial is with zero suppression thresholds set to 31, the maximum. The zero suppression will leave only the pre-sample data and thus every word count should be nine. A bit is set for any channel which fails.

At the same time the word count test is being done, the good dump counter is being tested. Since triggers cannot be aborted in this test system, the counter should increment with every trigger. The current value of the counter is read from channel zero and is thereafter tracked by reading from a different channel after every trigger. This is done until every channel has been checked. Each channel which fails will set its corresponding bit.

The 'valid data' bit for FEB event area zero is checked in each test. If its value is incorrect the corresponding bit is set and displayed. Normally, only event zero is used by our clock trigger board so it is the only one tested.

The 'zero suppress override' bit is set and each channel is checked to see that it did not suppress any data. This is checked only by examination of the word count.

Finally, the 'write to FEB' is started. This is just a memory test of the FEB space. This is a complete, but slow, duplicate of the test done by the microprocessor during its shakedown. Any keystroke will terminate this test early. The full test takes about ten minutes on the Ames setup.

3.6 Multi-scan Test

The simplest of the higher level tests is the Multi-scan test, which uses the microprocessor to collect data at each trigger. After all the data are collected in the microprocessor memory, it is read by the host VAX computer.

Each of these higher level tests has its own data format in the microprocessor memory. Consult the documentation in the listing of the microprocessor program (art$root:[M68K]-artemis.com) for detailed information. Each test produces exactly one block of data beginning at the address of the board event buffer. Every block begins with a word count followed by an identification word. Since there is only one common buffer for all the tests

in Artemis, this identifier word is checked to insure that each test has the correct type of data block.

Table 3.10 shows the terminal display used by this test.

Table 3.10. Multi-scan Display

```
+--- 23-NOV-88 10.59.45 #693 ----+
| +----- Multi scan  test -----+ |      +------- Basic Parameters -------+
| | G Multi scan  test         | |      |Channel number          0       |
| | D Display data             | |      |# Points/Scans        100       |
| | S Data Summary             | |      |Loop Delay             80 ms    |
| | A Add Summary to Log       | |      +---------------------------------+
| | L List data                | |
| | M Modify parameters        | |
| | F Function Generator       | |
| +----------------------------+ |
```

Multi-scan consists of one basic test. When Option G is selected the microprocessor is directed to accumulate a histogram for each channel on the board. Each histogram consists of 256 bins, one for each time bucket. The host computer then delivers a prescribed number of triggers to the system and entries are made in the histograms after each trigger. After all triggers are delivered the host computer reads the data and generates normalized histograms for each channel. These histograms are of the value of each time slot, averaged over many events.

Before starting a test, some parameters must be chosen. Option M is used to modify the parameters in the 'basic parameters' display. The channel number selection selects the first channel only for the display routines. Data values are always taken for all channels. The next parameter is the number of scans to average. The operator must be aware that the microprocessors uses only a 16 bit word for each bin of the histogram. This limits the number of scans that will function correctly.

The loop delay is the amount of time the host computer waits for the microprocessor to complete its program. This parameter is usually set once by the programmer and not modified. To select this time, run the test with a large number of scans and examine the programmed LED on the front panel with an oscilloscope. The LED is turned on and off as the program on the microprocessor starts and stops. The loop delay is decreased until a good duty cycle is achieved. Note that the smallest increment of time allowed on the VAX is 10 milliseconds. This loop delay is used in many of the complex tests. In most cases an acceptable value has been chosen as the default.

Option F will allow the operator to set the function generator. This is the same subroutine as used in the digitizer primitives.

Options D, S and L display the data in various formats. Option D is a graphic display, S is just the average value of each histogram, and L lists the data taken from the

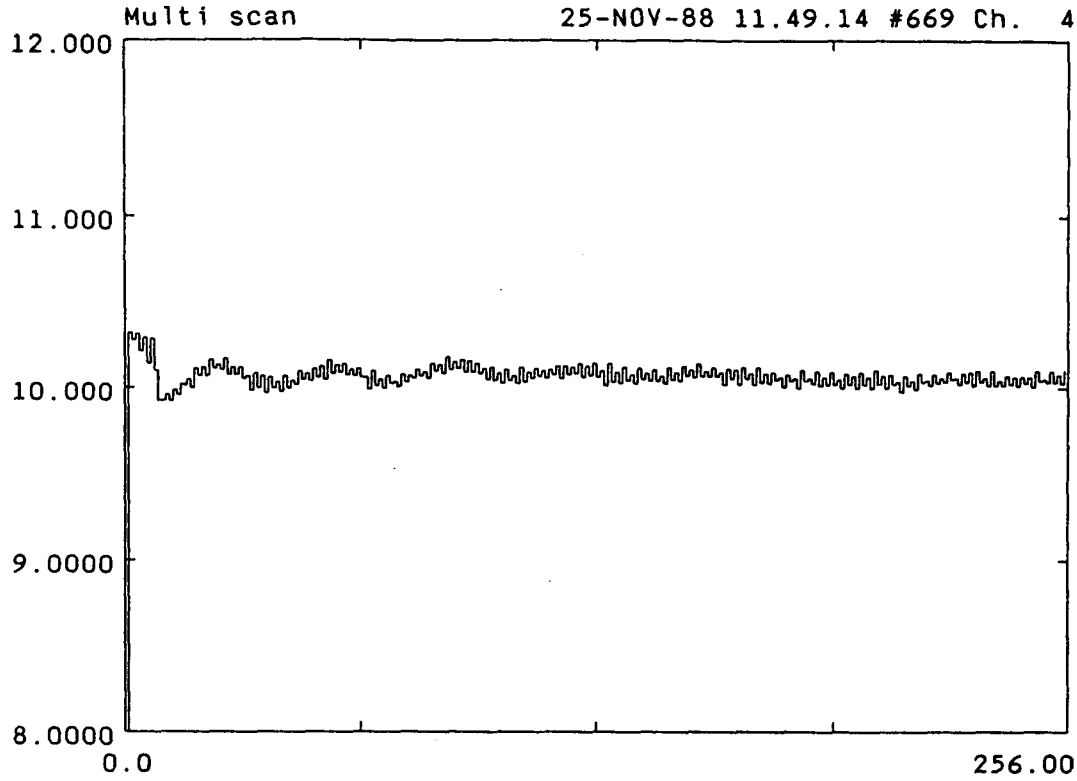

Figure 3.1. Output of Multi-scan Test

microprocessor.

Figure 3.1 is the result of a multi-scan run with 800 triggers. In this run the input was set to a sine wave with an amplitude of six millivolts and frequency of 500 kHz. The sine wave serves to prevent the ADC from locking into a single code. The average of many transitions from one code to the next provides a better measurement of the DC offset. A ringing effect, associated with the start of the scan can be seen. The high frequency (bin-to-bin) effects are most likely due to limitations of the FADC.

3.7 Linearity Test

The linearity test measures DC linearity of the analog buffer and Flash ADC combination. This test is primarily useful for checking the flash ADC tap voltages and the linearity of the analog buffers. Both linear and bilinear modes of the digitizer can be tested.

The linearity test causes the function generator to produce DC levels as input to the digitizer board and then triggers the system. The microprocessor will, in response to each trigger, compute an average (sum) for each channel and store these averages sequentially in the microprocessor memory. At the end of the test, the buffer is read out and various fits and displays are computed. The operator may then select the channel and display

Table 3.11. Linearity Testing Display

```
+--- 23-NOV-88 10.59.45 #693 ----+   +------- LinTest Parameters --------+
| +---- FADC Linearity testing --+-+ |Channel number          0          |
| | C Channel Number             | | |                                   |
| | D Display Data               | |No. Voltage Steps     250          |
| | F Fit summary                | |Break pt. (ADC's)    64.5          |
| | T Two Fit summary            | |Break pt. (Volts)    0.145          |
| | A Adjustment summary         | |Ripple Below BP        7. mV        |
| | G Edit function generator    | |Ripple Above BP       22. mV        |
| | L Linear Ramp Scan           | |Linear Fit Limits   0.000   2.150  |
| | M Modify Parameters          | |Voltage Range       0.000   2.150  |
| | R Record Two Fit Summary     | |Time Bucket Limits      5     250  |
| +------------------------------+-+ +-----------------------------------+
```

desired. Table 3.11 shows the terminal display used by LinTest.

Option M is used to modify the 'LinTest Parameters'. Again 'channel number' is only for the display routines. 'No. of voltage Steps' is the number of DC levels used. Internal arrays limit this to a maximum of 256 steps. 'Break pt. (ADCs)' is the breakpoint value (in ADC counts) used by the constrained fitting routine. 'Break pt. (Volts)' is used by the algorithm which computes the step sizes during the scan. The voltage step sizes are computed such that equal size steps in ADC values will occur. A slope ratio of 3.8:1 is assumed in this calculation. The break point value in volts will change with changes in the gain of the system. 'Ripple' is the amplitude of a sine wave added to the DC levels. This causes the FADC to output codes above and below the code corresponding to the DC level in order to remove some of the digital quantization effects from the output. Everyone should see the results of setting the ripples to zero at least once. 'Linear Fit Limits' are the upper and lower limits of the data used by the fitting routines. 'Voltage range' is the upper and lower limits of the generated DC levels. Changing the fit limits will change the results of the fits for existing as well as new data.

Option L begins the data collection. The function generator will display the current value of the DC offset. The time required to set the function generator is more than enough to allow the microprocessor to finish its program. Thus a loop delay is not needed.

Options F, T and A generate summary displays of the data for all channels. The 'fit summary' is the results of the fitting routine BIFIT. Because of the constraint on the breakpoint, the data from BIFIT are sometimes hard to interpret. The 'twofit summary' shows the results of TWOFIT, where no breakpoint constraint is used. The computed breakpoints are displayed with the fit results. See section 2.3 for more details on the fitting routines.

The 'adjustment summary' is an attempt to compute the desired change in one-quarter and full-scale tap point voltages. At these two tap points it is possible to insert trim resistors to adjust the response for each FADC individually, the goal being to compute

the optimum value for each FADC. Measurements show the changes in the quarter and full scale voltages are +6 mV per Ohm and -10 mV per Ohm, respectively. The current algorithm does not seem to work well. It uses the results of the routine FOURFIT (fit to each quadrant) and TWOFIT to compute the change in voltage required to make the fit to a single quadrant (second and fourth only) cross the fit to three quadrants at the center of each quadrant.

Option R records the results of the TWOFIT summary in a log file along with the serial number of the board and the timestamp. This will (hopefully) be the source of calibration constants, eventually. Naturally, these results are only worth saving if the board is in its final configuration.

Option D initiates the graphics display menu shown in table 3.12. This menu provides access to the data collected in LinTest.

Table 3.12. LinTest Display Menu

```
+--- 23-NOV-88 11.23.34 #693 ----+   +------- LinTest Parameters --------+
| +---- FADC Linearity testing --+-+ |Channel number          0          |
| | +------- Linear Scan --------+ | | |                                 |
| | | 1 Display Linearity Plot   | | |No. Voltage Steps     250          |
| | | 2 Display (Fit)-(Data)     | | |Break pt. (ADC's)    64.5          |
| | | 3 Display Diff Linearity   | | |Break pt. (Volts)   0.145          |
| | | 4 Display RMS Plot         | | |Ripple Below BP       7. mV        |
| | | 5 Display (Fit-Data)       | | |Ripple Above BP      22. mV        |
| | | E Edit plot parameters     | | |Linear Fit Limits   0.000   2.150  |
| | | C Change Channel           | | |Voltage Range       0.000   2.150  |
| | | F Toggle FITFLAG           | | |Time Bucket Limits      5     250  |
| +-+ L List Data on Screen    +-+ +-----------------------------------+
| H | M Modify parameters       |
| W | R Repeat LINFIT on Data   |
+---+-------------------------- Fitting Data ----------------------------+
|    23-NOV-88  11.23.34                      Voltages at:                |
|Slope 1 =     353.65  Intercept 1 =      11.27          0 FS = -0.032  |
|Slope 2 =      93.40  Intercept 2 =      50.56        1/4 FS =  0.151  |
|CHISQ   =       8.19                                  1/2 FS =  0.824  |
|#Voltage Steps = 250  Break Pt.    =       64.5        3/4 FS =  1.509  |
|Int. Volts =  0.151.  and ADC counts =    64.66          1 FS =  2.189  |
+-------------------------------------------------------------------------+
```

Options 1, 2 and 5 are the only plots currently implemented. The 'linearity plot' (1) displays the average ADC code as a function of input voltage, as shown in figure 3.2. The crosses are measured data points and the line is the result of using TWOFIT on the data points shown. This is the classic bilinear plot describing the FADC response.

Option 2 displays the difference between the data and a fit to the data (residuals) as a

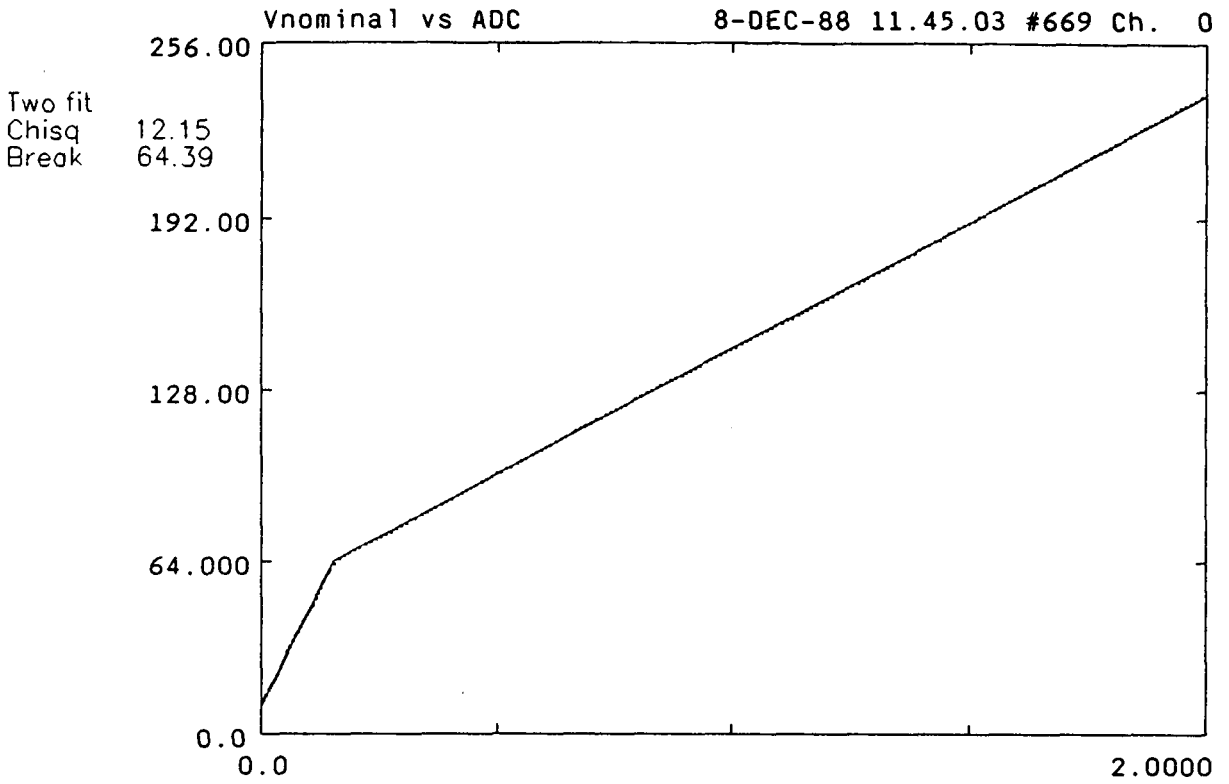

Figure 3.2. Direct linearity

function of voltage. Because this display squeezes the first quadrant into only about one twelfth of the horizontal scale, the plot on option 5 was created where the horizontal scale is the actual ADC code from the data. This is the most frequently used plot in LinTest. Figure 3.3 shows the plot generated by option 5 and the default setup shown in table 3.12. The fit shown is FOURFIT where each quadrant is fit to a line. Note that several points near each boundary are omitted from the fit. The step in the data near the center of the plot is due to a minor flaw in manufacturing the FADC.

Options E, C, L, M, and R operate as described except that TWOFIT is the only fitting routine used in LinTest.

The 'Fitting Data' display should also be revised. The slopes and intercepts are correct except the units are not shown. For those of us who have forgotten, they are ADC counts per volt.

3.8 Sawtooth Test

The 'SawTest' uses a sawtooth waveform to measure differential linearity. In a direct linearity, the overall deviation from a straight line is measured, but good direct linearity can be achieved with widely varying steps in voltage between consecutive ADC codes. Differential linearity measures the step size between each ADC code.

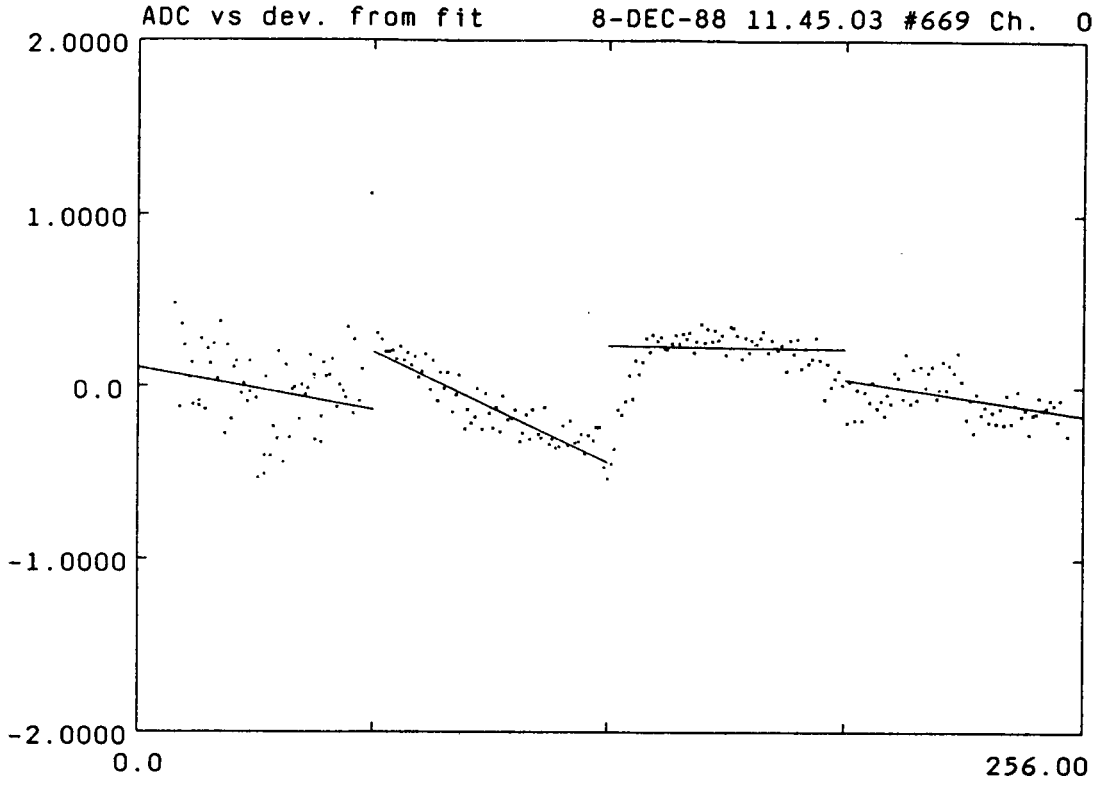

Figure 3.3. Deviation from Fit

To perform SawTest, the function generator is set (by the software) to a triangular function with a slow rise and rapid fall ('sawtooth') at 70 kHz. This frequency is sufficient to cause a complete cycle to occur within one trigger. The microprocessor identifies this single cycle and tallies how often each ADC code appears in the rising part of the data. In a perfectly linear flash ADC, each code should appear equally often because the linear ramp covers the entire range uniformly. In the bilinear case, data in the lower quadrant will appear less often due to the smaller step size (i.e., 5 mV/Count versus 19 mV/Count) in this quadrant. The program corrects for the variation caused by the bilinear response of the FADC before displaying the data.

Table 3.13 shows the menu display used by SawTest. Most of the options are very similar to those used in the linearity testing. Option S begins the collection of data.

Access to the function generator is provided because a change in the gain of the system may cause the programs to fail. The microprocessor program searches for both overflow and underflow as markers of the beginning and end of cycle. If the input waveform does not cause ADC codes zero and 255 to occur on every cycle, the program will tally the wrong data.

Figure 3.4 is the final normalized differential linearity plot. A value of zero indicates that the corresponding code (x-axis) is exactly as wide (in mV) as the average voltage step

Table 3.13. SawTest Display

```
+--- 23-NOV-88 11.27.14 #693 ----+
| +---- FADC SawTooth testing ---+-+  +------ SawTest Parameters ------+
| | C Change Channel               |  |Channel number          0       |
| | D Display data                 |  |No. Sawtooth Scans    100       |
| | G Edit Function Generator      |  |Break pt. (ADC's)  64.500       |
| | L List Data on Screen          |  |Break pt. (Volts)   0.320       |
| | M Modify parameters            |  |Vref +    (Volts)   4.000       |
| | S SawTooth Scan                |  |Low  bin No.(IWLO)     10       |
| | T Summary of D.Lin             |  |High bin No.(IWHI)    250       |
| +--------------------------------+-+  |Loop Delay              60 ms    |
| S SawTooth Test                  |    +---------------------------------+
| G Delta test                     |
| H Multi event scan               |
| +-------------------- Calculated Parameters ---------------------+
| |                     slope1   int1     slope2   int2     chng    |
| | Nominal values    219.130   -6.862   45.697   49.220 ........   |
+-+ Adjusted values   185.626  -50.278   52.684   32.282 ........   |
  +-----------------------------------------------------------------+
```

Table 3.14. SawTest Display Menu

```
+--- 25-NOV-88 12.54.03 #669 ----+
| +---- FADC SawTooth testing ---+-+  +------ SawTest Parameters -----+
| | +------ SawTest Displays ------+-+|Channel number          4      |
| | | 1 Raw Tally                  | ||No. Sawtooth Scans    100      |
| | | 2 Differences                | ||Break pt. (ADC's)  64.500      |
| | | 3 Normalized Tally           | ||Break pt. (Volts)   0.320      |
| | | 4 Differential Linearity     | ||Vref +    (Volts)   4.000      |
| | | 5 Code Centers               | ||Low  bin No.(IWLO)     10      |
| | | 6 Integral Linearity Error 1 | ||High bin No.(IWHI)    250      |
| +-+ 7 Integral Linearity Error 2 | ||Loop Delay             60 ms   |
| S | E Edit plot parameters       | |+-------------------------------+
| G | C Change Channel             |
| H +------------------------------+---+
```

size. A value of minus one indicates the code never appears in the output and the voltage step size is zero. A value of plus one indicates that the step size is twice as large as the average. The differential linearity can vary depending on the speed at which the FADC is operated and the frequency of the input signal.

An interesting feature in this plot is the high bin at the ADC code 65. This is at the point where the slope changes by a factor of about four. Data values to the left of this point have been scaled differently from the data values to the right. The 'dip-spike'

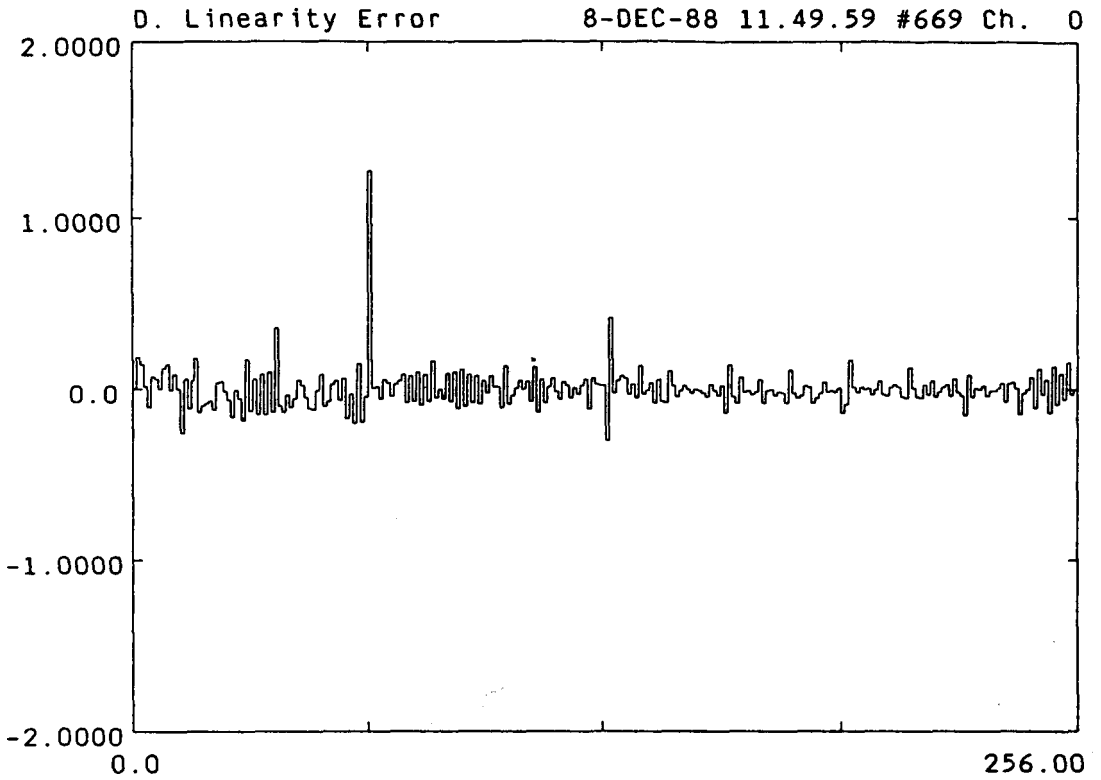

Figure 3.4. Differential Linearity

feature in the center of the plot shows that some data which should have been code 127 appear as 128.

3.9 Pulse Test

The Pulse Test is similar to LinTest except that the amplitude of a pulse is measured rather than a DC level. This allows the measurement of linearity through some of the front end electronics which are not part of the digitizer module itself and are ac-coupled. In this test it is also possible to use a LeCroy 9100 Arbitrary function generator in place of the HP function generator. If the LeCroy 9100 is used, any waveform previously loaded can be used, for example, a pulse with a 1/T tail. The HP will use a single cycle of a sine wave, with the baseline set to the minimum value of the sine wave, to form a pulse. It is also possible to use a square wave as input to a differentiation circuit to generate the input pulse.

In this test the electronics must be set up to cause the function generator to trigger with the system. In the Ames test bench a delay box is used to delay the synchronous trigger from the clock trigger board to the function generator. This allows the centering of narrow pulses on a single time slot in the readout.

Table 3.15 is the display shown by Pulse Test. Many of the options are similar to

those of LinTest. Note that the 'ripple voltage' of LinTest is gone. This is usually not necessary once the digital effects are understood.

Table 3.15. Pulse Test Menus

```
+--- 23-NOV-88 11.27.14 #693 ----+
| +---------- Pulse test --------+-+   +------- Basic Parameters -------+
| | P Pulse test                 |   |Channel number          0       |
| | S Pulse test(68k)            |   |# Points/Scans         25       |
| | R Voltage Ramp               |   |Low  bin No.(IWLO)     65 bkt   |
| | F Fit Summary                |   |High bin No.(IWHI)     70 bkt   |
| | T TwoFit Summary             |   +--------------------------------+
| | D Display data               |
| | L List data                  |   +------- Pulse test data --------+
| | C Change Channel             |   |Pulse height avg    44.00 cts   |
| | M Modify Parameters          |   |Pulse height rms     0.00 cts   |
| | O Modify Ramp                |   |Pedestal avg        11.00 cts   |
| | G Pulse generation           |   |Pedestal rms         0.00 cts   |
| +------------------------------+-+   +--------------------------------+
| Z Snapshot (any menu)          |     +--------- Voltage Ramp ---------+
| Q Quit (any menu)              |     |Lower Voltage          5. mV    |
+--------------------------------+     |Upper Voltage       2150. mV    |
                                       |No. of steps          100       |
                                       |Break voltage         145. mV   |
                                       |Loop Delay              0. ms   |
                                       +--------------------------------+
```

One display, not in LinTest, is the 'Pulse Test Data' which shows the result of some number of scans at a single amplitude. This is useful to see that the electronics are set correctly. Note that the microprocessor program will search for the pulse only in the range specified. Thus a narrow range will reduce the required time to execute the program and hence the necessary loop delay. The pedestal is computed from five consecutive time buckets beginning ten buckets away from the peak. Thus, very wide pulses will not work correctly. The pedestal is computed and displayed but not subtracted from the peak value or used in any way.

Options P and S generate data for a single amplitude. Option P does not use the microprocessor and is thus very slow. Option R does a ramp of the amplitude in LinTest fashion. The generated data are put in the displays.

Option G is used to set up the pulse used in the test. With this menu the switch to the LeCroy 9100 can be made. This option is general enough to permit arbitrary GPIB commands to be sent to either function generator.

Figure 3.5 shows the residuals from a pulse test. Note that the regular pattern of deviations from the fit is the result of the digital quantization by the FADC. This effect

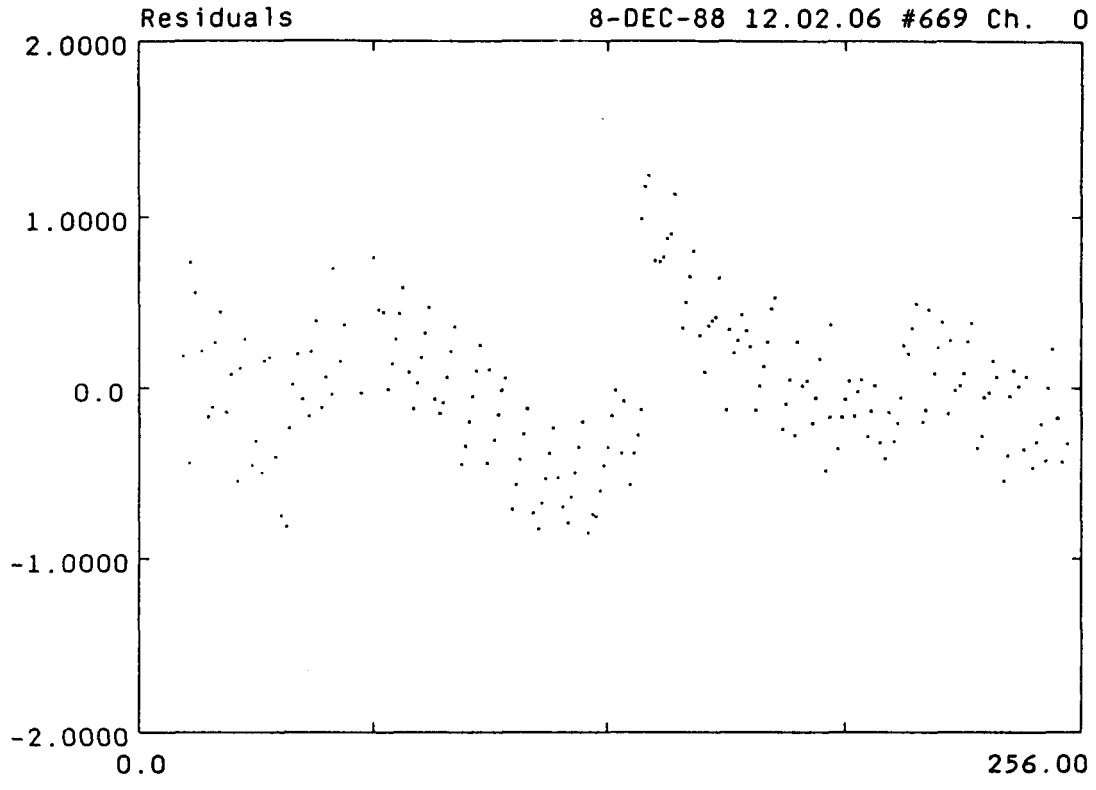

Figure 3.5. Pulse test Results

can be clearly seen in LinTest by setting the 'ripples' to zero.

3.10 Delta Test

One of the common flaws in the performance of the FADC is incorrect codes in the output. Delta Test is a powerful tool to search for the occurrence of this problem at very low levels. The Delta test generates a histogram of the differences between consecutive values in the data. Thus if an input signal is chosen in which only ADC steps of one count are expected in the output, a step of two counts is an error. Table 3.16 is the display used by Delta Test.

To run Delta Test an input wave form is chosen and the number of scans and the range of time slots is set. Note that the pre-sample data must not be included in the range of time slots. Then the test is initiated. In about twenty minutes, ten thousand scans can be executed. This corresponds to about 2.5 million ADC conversions per channel examined by the microprocessor.

Figure 3.6 is the result of a Delta Test for a single channel. The input wave form is a triangle wave at 250 kHz with an amplitude of 200 mV. The offset is set to 830 mV which makes the average code 128. This waveform should never require a step in the output of more than one ADC count. Thus the single datum in the wings is probably the result of

Table 3.16. Delta Test Displays

```
+--- 23-NOV-88 11.27.14 #693 ----+
| +------- Delta  test --------+ |    +------- Basic Parameters -------+
| | G Delta  test              | |    |Channel number          0       |
| | H Delta search             | |    |# Points/Scans        100      |
| | D Display data             | |    |Low  bin No.(IWLO)     10 bkt   |
| | S Data Summary             | |    |High bin No.(IWHI)    250 bkt   |
| | L List data                | |    |Loop Delay             80 ms    |
| | M Modify parameters        | |    +--------------------------------+
| | F Function Generator       | |
| +----------------------------+ |
```

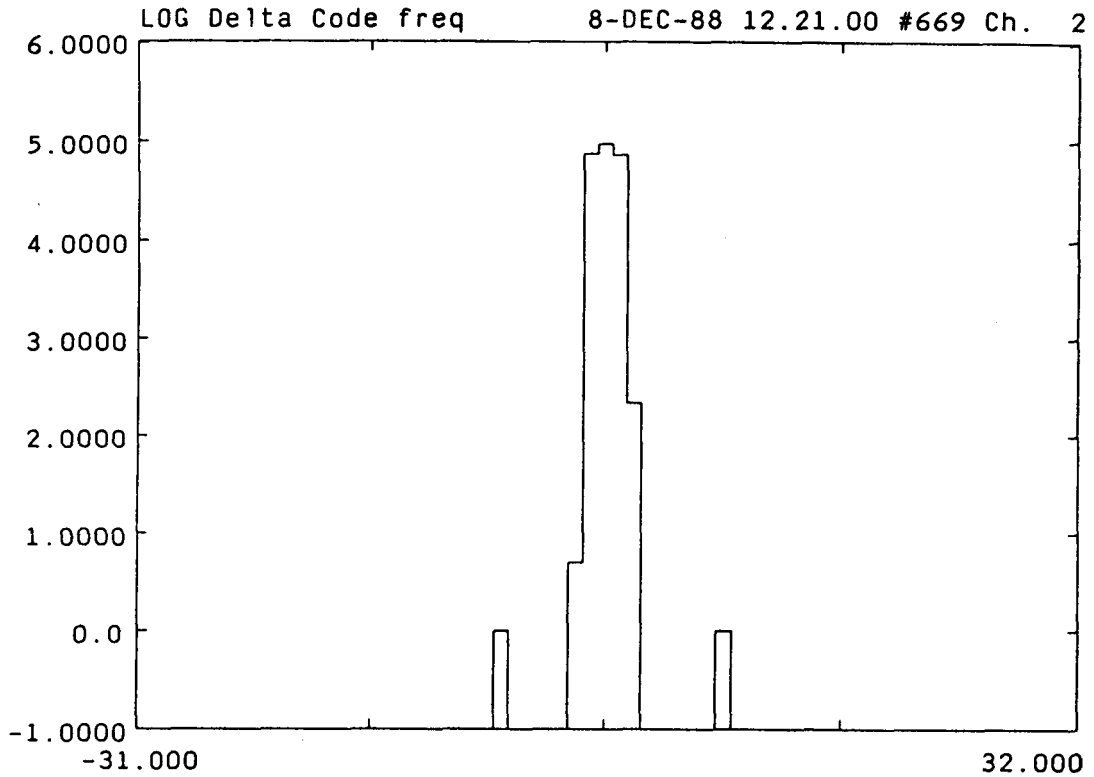

Figure 3.6. Delta Test Results

an incorrect bit (03) in the output of a single time slot. The center bin represents about one hundred thousand steps of zero ADC counts.

The data summary lists the log of the sum of the contents of the center three bins of the histograms (steps 0, +1, and −1) divided by the sum bins for steps sizes greater than 1, 2 or 5 (plus one to avoid divide by zero). Thus if the number displayed is 4.0, the error occurs at a rate of less than 1 in 10^4.

Option H provides a search for errors. While the test may count errors quickly, an

example of the error may sometimes be hard to find. This option will execute Delta Test until an error occurs on the channel shown in the 'basic parameters' display. This is much slower than the actual test since the result of each trigger must be checked by the host. When the search terminates on an error (termination will also occur when the number of scans indicated is reached) the operator must go to the digitizer primitives to read data without triggering the system, in order to see the error. The operator could also examine other channels to see if the error extends beyond the single FADC. This would indicate a different type of problem.

3.11 Bookkeeping

The bookkeeping options allows the user to enter notes regarding the board or FADCs being tested. Table 3.17 is the menu displayed by the bookkeeping module.

Table 3.17. Bookkeeping Menu

```
+--- 23-NOV-88 11.27.14 #693 ----+
| +--------- Bookkeeping --------+-+
| | E Edit Log   693              |
| | F Edit Log     0              |
| | L LDUMP                       |
| | V VAX DCL                     |
| | A Add data FADC Stats         |
| | D Display FADC Stats          |
| +-------------------------------+-+
```

The log file is maintained as a simple text file. The entry for each board begins with the line "Digitizer #nnn". Where 'nnn' is the serial number of the board. When option E or F is selected, the EDT editor is entered and this header is found. Next the string "Comments:" is found. After this string the current time and date are written, followed by the board's serial number. The cursor is placed on the next line and the operator may type any text desired. After entering all the information the editor may be exited and the log file updated by the standard EDT 'exit' sequence. If you wish to leave the contents of the log file unchanged, use the EDT 'quit' sequence, instead. Other modules can enter data in this log file. The best example is LinTest. In LinTest all of the current TWOFIT results are recorded as if the operator had typed them in.

DCL commands are also available. This avoids the necessity of exiting the test bench and losing collected or entered data. Option L executes the command file LDUMP to print the saved plots and displays.

Options A and D were added to study samples of FADC. In this case the results of FOURFIT in LinTest are recorded according to the FADC production lot numbers in separate log files.