Calibrating CLAS FASTBUS TDCs

Robert Millner And Elton Smith
Sept. 27, 1993

It is necessary to test, calibrate, and determine the conversion parameters of the FASTBUS TDCs that will be used in the CLAS TOF system. Specifically, it is necessary to ensure that the TDCs conform to the CLAS requirements; that the RMS resolution of each channel is less than 50ps, the dynamic range is greater than 150ns, and that the enable time is less than 35ns[A]. The conversion parameters from TDC output counts to actual time must also be determined. This is done by applying a quadratic fit to the input data. Furthermore, as this will be applied to many modules, several input delays and for large sets of data, the procedure must be semi-automated. To that end, several pieces of software and hardware were developed to run with the CODA (version 1.2) system on our DEC 5000. A standard procedure was also developed and applied to a LeCroy 1872A TDC prototype module for the CLAS. This document is a guide for testing and calibrating the TDCs, as well as a summary of specific tests that were conducted.

Recommended Reading

CLAS-NOTE-91-022, "FASTBUS TDCs for the CLAS TOF", by Elton Smith and Shouben Zhou.
CODA manual (version 1.2)
CLAS-NOTE-93-016, "CAMAC Through VxWorks", by Robert Millner

1 Hardware Setup

Refer to Figure 1 for an overview of the apparatus and setup. Testing and calibrating a TDC requires access to a programmable source of delay signals that is more accurate than the device being tested. To that aim, a CAMAC Phillips 7120 Charge Time Generator (QTG) was used in a standard CAMAC crate which was being controlled by a Motorola MVME 162-12 attached through a K-Bus interface. Care must be taken to synchronize the signals to the QTG and the FASTBUS TDC. Otherwise the data collected would be invalid. Finally, the FASTBUS under the control of a Fermilab FSCC had to receive a trigger to cause it to readout the TDC. The FSCC could not receive the next trigger until it was finished reading out the previous event or else the new event would never be read out. To this end, several modules of NIM logic were used to control and inhibit QTG signals depending on the status of the FSCC.

1.1 CAMAC Setup

The CAMAC Phillips 7120 QTG calibration module uses a 16 bit DAC to produce test signals on a range of 0 to 512 ns. Its counts to time conversion[B] is linear with no offset and a linear coefficient of 7.8 ps/count with an expected jitter of 30 ps. This was placed in slot 7 of the

CAMAC crate. Although the slot number is irrelevant in the hardware, if a different slot is used, then the program RunCamac.exe must be modified[1]. A standard test run to this module involves stepping from a minimum to a maximum range of output signals several times. The QTG will output a signal approximately 100 μs after the 16bit DAC has loaded it. The signal will be output in the form of two lines, one from the "start" and one from the "stop." Both lines are BNC connectors which will produce -32 mA, 50ns pulses with rise times of less than 1.5ns. Both connectors are back-terminated with 50Ω impedances. The module is linear to 11 bits and accurate to +-2% of the selected full range[C].

A LeCroy 2228A TDC was placed in slot 10 in order to provide the CAMAC system with a Look_At_Me (LAM) when the FSCC is ready to receive a new trigger. As with the QTG, the slot number only matters to the RunCamac.exe program. The line delivering the pulse must be insert into the "COM START" input. This will activate the module when a signal is transmitted. After activation, the module will wait until it times out and set the LAM. No usable data other than the LAM is returned from the TDC in this setup. The START input is a 50Ω terminated Lemo connector and requires a NIM level signal[D].

The CAMAC crate is interfaced through a Kinetic Systems 3922 Parallel Bus Crate Controller. The controller has a selector for the crate number. Presently it is set to zero. The Controller also has a switch to place the crate on-line or off-line and a switch to clear and reset the crate. Clear and Reset functions from the switch require that the crate be taken off-line. There are five status lights on the front face of the controller. The busy light should activate whenever the crate is accessed, the no Q or X on an error condition. The inhibit light signifies that the crate is inhibited. This should be off during a run. The SLP light should also not be active during a run. There are four Lemo connectors on the front panel. The "busy" and "grant out" connector should be left open. The request and grant in should be connected together to allow the crate to be accessed.

The Crate controller is interfaced through a Kinetic Systems 2917 K-Bus interface on the VME crate which is controlled by a Motorola MVME162-12 VME Controller. The MVME162 provides a terminal and network interface to its functions and CODA with the VxWorks operating system. A twisted pair serial cable is attached to the "SERIAL PORT CONNECTOR" on the front of the MVME162. A Milan MIL-02T Thinnet connector attached to the "ETHERNET" port in the back provides access to CEBAF Ethernet which will carry the data from our DEC station. The control program, RunCamac.exe uses the ethernet to transmit information to and from the MVME162. The MVME162 can be rebooted by pressing the "reset" switch on the front panel. This should be done once before starting an experiment. For more information on CAMAC, refer to CLAS-NOTE-93-016, "Talking to CAMAC Through VX Works."

1.2 FASTBUS Setup

The LeCroy 1872A TDC has 64 channels each of which can digitize with 12 bits of accuracy. The autoranging feature was disabled in this module as it was unnecessary. This device was set to count 50 ps per increment giving it a possible range of 204ns[G]. The common start and stop inputs are differential ECL with 110ohm impedance and a minimum activation width of 10 ns. One of the characteristics to be measured is the minimum delay between the start and stop signals

1. See appendix A.

that is needed to activate the channel (enable time). The 64 channels are spaced along the front of the module in groups of 16. This makes it convenient for a ribbon cable to be attached to the front end from a fan out and an NIM to ECL converter.

The FASTBUS crate is controlled by a Fermilab FSCC PC4 module running VxWorks. This module provided access to the ethernet which is used to transmit the data to CODA. In addition to ethernet, a terminal may be connected on the front panel through the "TERM" connector[1]. A trigger signal is needed in order to readout a module. This is provided by a differential TTL signal through a twisted pair ribbon cable, pin 6 on the front panel input port. A "done" pulse is transmitted from pin 1 of the output port and signifies that the FSCC has completed its readout. Any triggers received while the FSCC is busy will be ignored, so this signal is used to release the inhibit on the logic when the controller is not busy.

1.3 NIM Logic

The NIM logic was used to coordinate the delivery of CAMAC generated start and stop signals to the TDC with the trigger to the FSCC. In order to insure that all the transmitted events were delivered properly, the FASTBUS done signal is used to issue a LAM to CAMAC indicating that a new start/stop sequence may begin. If this loop is not properly timed events will be lost and the run will not have synchronized data. Refer to Figure 2 for a simplified version of the setup. The chain of events was started by the QTG delivering a start/stop pair. The stop pulse is fanned out to 16 channels in a LeCroy 429A Logic Fan Out. The stops are converted to ECL in a Phillips 726 Level Translator and transmitted over a ribbon cable 16 channels at a time to the TDC. The start signal is fanned with a Phillips 757 Mixed Logic Fannout to a scaler and a Phillips 755 Logic (AND) Unit. The AND gate output to a Phillips 794 Gate/Delay Generator in flip-flop mode which enables the veto to the AND module. This prevents further signals from coming through. The AND fanns out the start to a second scaler, an ECL converter and to another gate/delay generator in delay mode. ECL then sends the start signal to the TDC. In our setup there is a considerable time difference, about 35ns, between the output on the QTG and the input on the TDC. This must be measured for each input and will be used to modify the calibration in the Calibrate.exe program. The second gate/delay generator is used to delay the trigger before reaching the FSCC until an event has been converted. This is set to 0.5ms which is considerably longer than the expected conversion time. The trigger pulse then goes into a third gate/delay generator set in flip-flop mode. This causes the "gate" signal to go high and the "not-gate" signal to go low which is translated to TTL in a Phillips 726 Level Translator and sent to the FSCC trigger input. The trigger line provides a level to the FSCC which may be polled by the software to determine when to readout the TDC.

The done output produces a signal when the FSCC is finished reading out the TDC. This is in the form of a noisy pulse that goes through the Phillips 757 Mixed Logic Fanout. The pulse then goes into an "AND" gate which has a programmable output length. The output length, 750ns, must be long enough to cover the entire pulse. This output is the "cleaned done pulse" as it has been converted into a long square wave. This pulse is fanned out to reset both flip-flops and to set the LAM. Clearing the flip-flops will reset the old trigger level and allow a new trigger to be

1. This serial I/O line expects a terminal that is running at 9600 baud, 7 data words and no parity.

delivered. The LAM tells the CAMAC software to deliver the next event, thus completing the loop.

2 Software Setup

The software in this experiment must fulfill three purposes (see Figure 3). First, the CAMAC driver software initiates signals in a predetermined sequence which is recorded in a file. This is done by the RunCamac.exe program. Second, the FASTBUS TDC events are acquired using CODA. Third, the FASTBUS data is analyzed by comparing it to the predetermined sequence generated by CAMAC. This is performed by the calibrate.exe program. This system is designed to test one device at a time independent of slot location and specific type of TDC. All types of TDCs to be used by CLAS are readable by the CoEvb_unpack[E] library and placed in a standard format. This removes any further dependency of the analysis program on the hardware. In the present version, the CoEvb_unpack library can decode both the LeCroy 1872A and Phillips 10C6 TDCs. Other TDCs can be decoded by adding them into the CoEvb_unpack library. See appendices A through D which describe each piece of software in detail.

The software driving this experiment must work together with the hardware in order to provide a synchronized event stream. Specifically, the RunCamac.exe program (see Appendix A) and the Readout Controller program (CRL) (see Appendix E.3) loaded onto the FSCC must provide signals to each other through hardware. This is implicit in the CRL file with the trigger and done routines. The RunCamac.exe outputs through the QTG and receives feedback through the LAM on the CAMAC TDC. The RunCamac.exe program initializes the loop by producing the first signal, which should be done after CODA has received a "go." The trigger bit causes the CRL to execute its trigger subroutine which block reads and clears the TDC. The trigger subroutine then calls a done subroutine which delivers the done bit through the FSCC. That done bit activates the LAM to trigger the next event. There must be a 1:1 correspondence between trigger and done bits or else this cycle will be incorrectly timed and the data will be invalid. There can be as much as a five second delay from trigger to done and still keep the experiment active. Longer delays will be interpreted as an error by the RunCamac.exe program and it will try to re-send the event. There is no time limit from done to the next trigger. The maximum event rate we obtained on our system was 9.5events/second in this setup. Clearly, a logical upgrade would be to convert the RunCamac.exe program to a CRL file on the CAMAC Readout Controller (MVME162). This should occur as soon as the event builder is capable of properly synchronizing both controllers.

The calibrate.exe program (see Appendix C) is used after data has been acquired to calculate statistical information on the device being tested. This program uses three input files; the RunCamac.exe output and the CODA Raw Physics Event file and the offsets file. The first two files contain event information and are required by the program. The third file contains channel by channel information of the time offsets induced by the setup between the start and stop pulses generated by the QTG. It is simply a text file with one real number on each line. These numbers are used in calculating the actual time delivered to the TDC from the QTG generated time. These offsets may be measured with an oscilloscope (see Section 3.1 below). If this file is missing, then the offsets are assumed to be zero.

The program produces two useful output files, the HBOOK Postscript Metafile and a table of fit parameters. The fit parameters are on records of length 1024 and format

"(x,i3,x,f9.5,x,G14.7e2,x,G14.7e2,x,f9.5,G14.7e2)." The first integer is the channel number. The next three floating numbers are the constant, linear and quadratic terms of a quadratic fit for that channel. The final two are the constant and linear terms of a linear fit. The constant terms are a measurement of the enable time. The postscript file contains a page of output for each channel and a summary of the fits at the end. An example of these pages are included as Figures 4 and 5. They are intended to be a qualitative representation of the performance of the TDC. From these outputs, a good idea of the enable time and linearity of each channel may be obtained. The usable range of the TDC and fits may also be obtained as well.

The appendices contain the documentation on all related pieces of software except for the CoEvb_unpack library and the CODA which have documentation of their own. The CoEvb_unpack library can be found in CLAS-NOTE-93-018. CODA has its own documentation. Copies of the CODA network, configuration and CRL files are included in appendix E as examples. There is one additional piece of software included in this write-up, the JoinRun.exe program. Its purpose is to concatenate several CODA Raw Physics Event files into one long Physics Event File. Depending on the setup, it may not be possible to test all of the channels of one device in a single run. The JoinRun.exe program allows all of the runs for one device to be packaged together for the convenience of having one event file with all of the channels of one device. It is up to the user to concatenate the files produced by the RunCamac.exe program. This can easily be done with a text editor or with the "cat" command. Care must be taken in order to ensure that the RunCamac.exe output files are concatenated in the same order as the event files in order to maintain synchronization between the two.

3 Sample Test Run

This is a quick overview of how to preform one test run. It is designed to give the user an idea of how the elements of this experiment work together and to assist in preforming tests.

3.1 Hardware

Please refer to Figures 1 and 2 for a graphic map of the setup. Place the Phillips 7120 QTG in slot seven of the CAMAC crate. Switch its "Primary DAC" to "T" and "TIME" to "CAMAC" and "512ns". Place the CAMAC TDC in slot ten of the TDC. Then set the crate number to zero on the front of the Parallel Bus Crate Controller and place it on line. If for some reason those defaults cannot be used, the RunCamac.exe program must be changed to look for the crate, QTG and TDC LAM in the different configuration. Take the stop output of the QTG, fan it out, convert it to ECL, and run it to the stop inputs of the FASTBUS TDC. Take the start output of the QTG and fan that out as shown in the Logic diagram. Convert it to ECL and place it in the common input of the FASTBUS TDC.

Place the FASTBUS TDC in slot 17. If that is not possible, than the CRL file must be changed to use the new slot. Place the FSCC in any available slot. Take the done output, convert it to NIM and clean up the signal to a single pulse. Fan out the clean signal as shown in the logic diagram. The signal to the LAM should be at least 50ns to ensure proper firing of the LAM.

Prepare the logic circuit as shown in the logic diagram. This is not intended to be a blueprint of how the logic was set up; rather a guide to the logic itself. Ensure that the signal from the

FSCC is a single square pulse coming out of the fan-out. Ensure also that the trigger signal to the FSCC is a differential TTL level. This may require customized cables. The trigger delay is provided to allow the TDC time to convert the signal to digital. The allowed time in the logic diagram is considerably larger than the necessary delay. Since the dead time produced by the CAMAC loop is in the order of a tenth of a second[1], 0.5 ms is a small time to add to completely ensure the proper arrival of a trigger signal. The Flip-Flops turn on the trigger and lock out extra triggers from arriving. Since the trigger signal is on until the flip-flop is reset, it is active until the FSCC delivers a done signal. This allows polling mode to be used in the CRL program. The two scalers are provided as an extra level of error checking. The value of the first is the number of signals generated. The second is the number of signals allowed into the logic. The number of triggers and dones should match the second scaler. If the two scalers do not match, then an error has occurred and the run is probably not synchronized. If the data is not synchronized, then the run is not valid and the Calibrate.exe program will not be able to give the proper statistics.

Once the hardware has been set up, remove the outputs of the QTG. Set the "TIME" to "MAN" and take the crate off line. Use a small screwdriver to give it some output time, the press "Free Run." This will cycle the QTG at 10KHz. Attach an oscilloscope to the start and stop outputs and measure the time differences. The time should be measured from the leading edge of the start to the leading edge of the stop, or the trailing edge of each. However that is done, it must be measured from the same place on each signal. To stop the QTG, click reset on the Bus Controller. Replace the start and stop inputs. Remove the veto from the "AND" gate for the trigger signal. Then press "Free Run" on the QTG. Now measure the time difference at the ends of the ECL cables for each output. Subtract the first value from these values and place the result in the calibration offsets file. Make sure that there are at least as many numbers in the file as channels that the calibration program will work with. Reset the QTG and place it in "CAMAC" mode. Place the crate on-line, then replace the veto input. Reset both flip-flops. The circuit should now be ready to take data.

3.2 FASTBUS and CODA

Make sure that the FASTBUS crate is on and functioning properly. Press reset on the front panel of the FSCC. This will reboot the controller and clear the crate. While it is booting, bring up RunControl and configure it. Make sure that all of the necessary components for CODA have ben prepared[2]. When the FSCC boots, press download on RunControl[3]. This will transmit the CRL program's object code[4] to the FSCC as well as enable several components. Then issue a prestart and a go to RunControl. The FASTBUS and logic are ready to receive signals from CAMAC.

3.3 RunCamac.exe

Execute the RunCamac.exe program. Use the menu selections to prepare the output loop. These selections have been set to run the TDC through its entire range, stepping every two nanoseconds for several iterations. These defaults work with the timing on our setup. If your setup dif-

1. Measured by triggering the LAM with the start output of the QTG in a self-loop.
2. See appendicies E.1, E.2 and E.3
3. If this sequence is unsuccessful, reboot the FSCC, then reset RunControl.
4. See appendix E.3 for a sample CRL file

fers considerably, the program can be reconfigured to bring up a different list of defaults by modifying the source. When properly configured, press enter. RunCamac should begin sending events and RunControl should begin registering events. If this is not the case, then there is an error in the setup. At the end of the run, RunCamac.exe will terminate itself and print messages on the number of events delivered. End the run by pressing end on RunControl. This will save the event file and stop the FSCC. The number of events reported by the two scalers and RunCamac should match. RunControl should report back that number plus three events. If these numbers do not agree there may have been an error in the run. RunCamac will report and attempt to correct some types of errors; however the data should still be considered useless.

3.4 Calibrate.exe

Once the data has been taken, activate the calibrate.exe program. This will present the user with a menu of options. The default selections for the input filenames match the default output filenames of the RunCamac.exe program and the event file. The metafile viewing option should not normally be used. After the defaults have ben set, press enter to begin the program. This program may be placed as a background task for the rest of its run. Since it is producing a postscript metafile, it will take some time to operate. Once it is finished, the metafile can be viewed or printed out on a postscript printer. The fit constants file may be used to convert the counts output data on the TDC to real input time. The calibration.exe program should not encounter an error during processing. If it does, try it on several different sets of data to see if the error is consistent. See Figures 4 and 5 for samples of the output data. Only channels with data will be output.

Figures

Racks For Cosmic Ray Test.....1

Logic Diagram.....2

Calibration Procedure.....3

Sample TDC Output.....4

Sample Fit Summary.....5

Figure 1

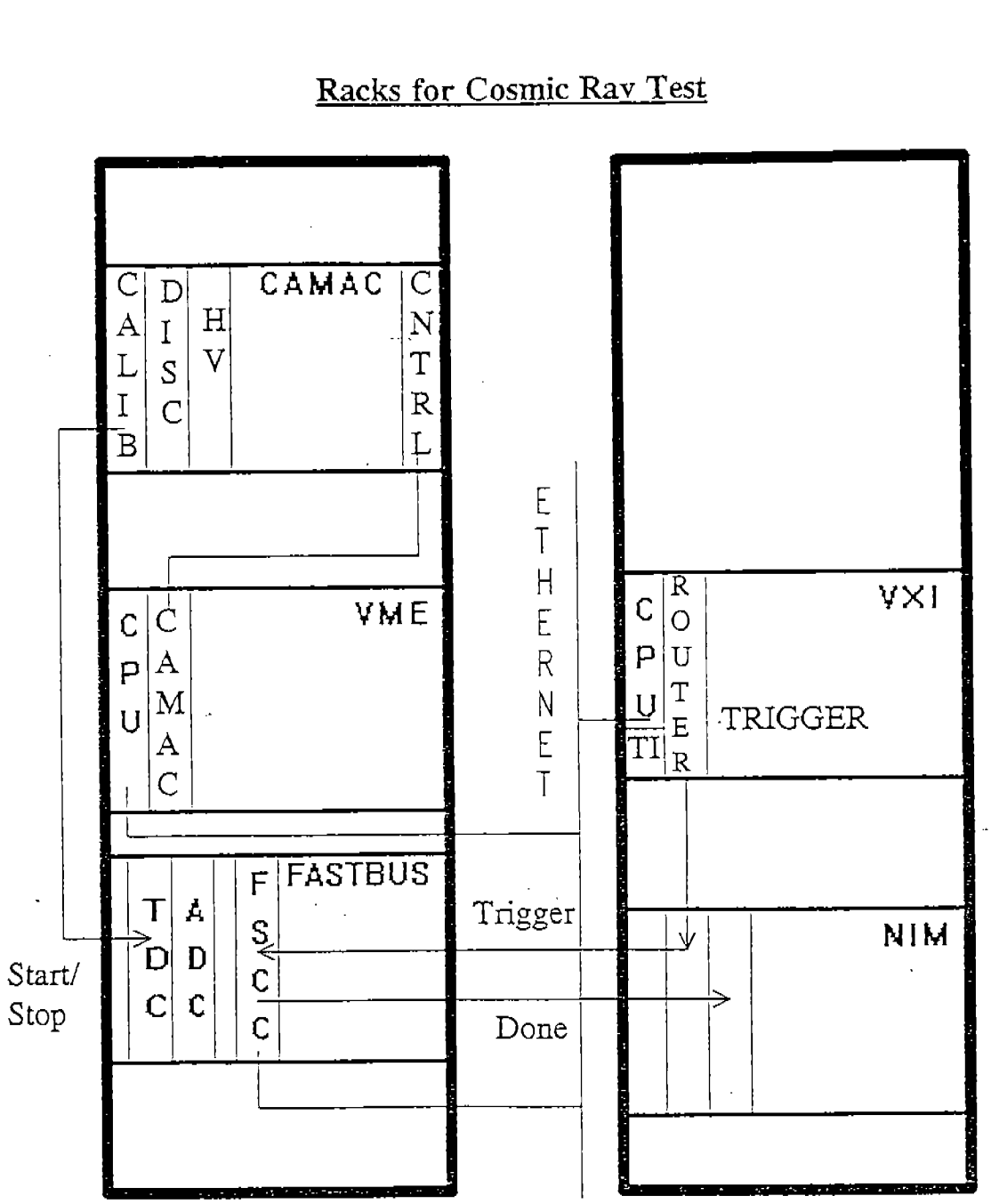

Racks for Cosmic Rav Test
CALIB
DISC
HV
CAMAC
CNTRL
CPU
CAMAC
VME
ETHERNET
CPU
ROUTER
TI
VXI
TRIGGER
FASTBUS
TDC
ADC
FSCC
Trigger
Done
Start/
Stop
NIM

Figure 2

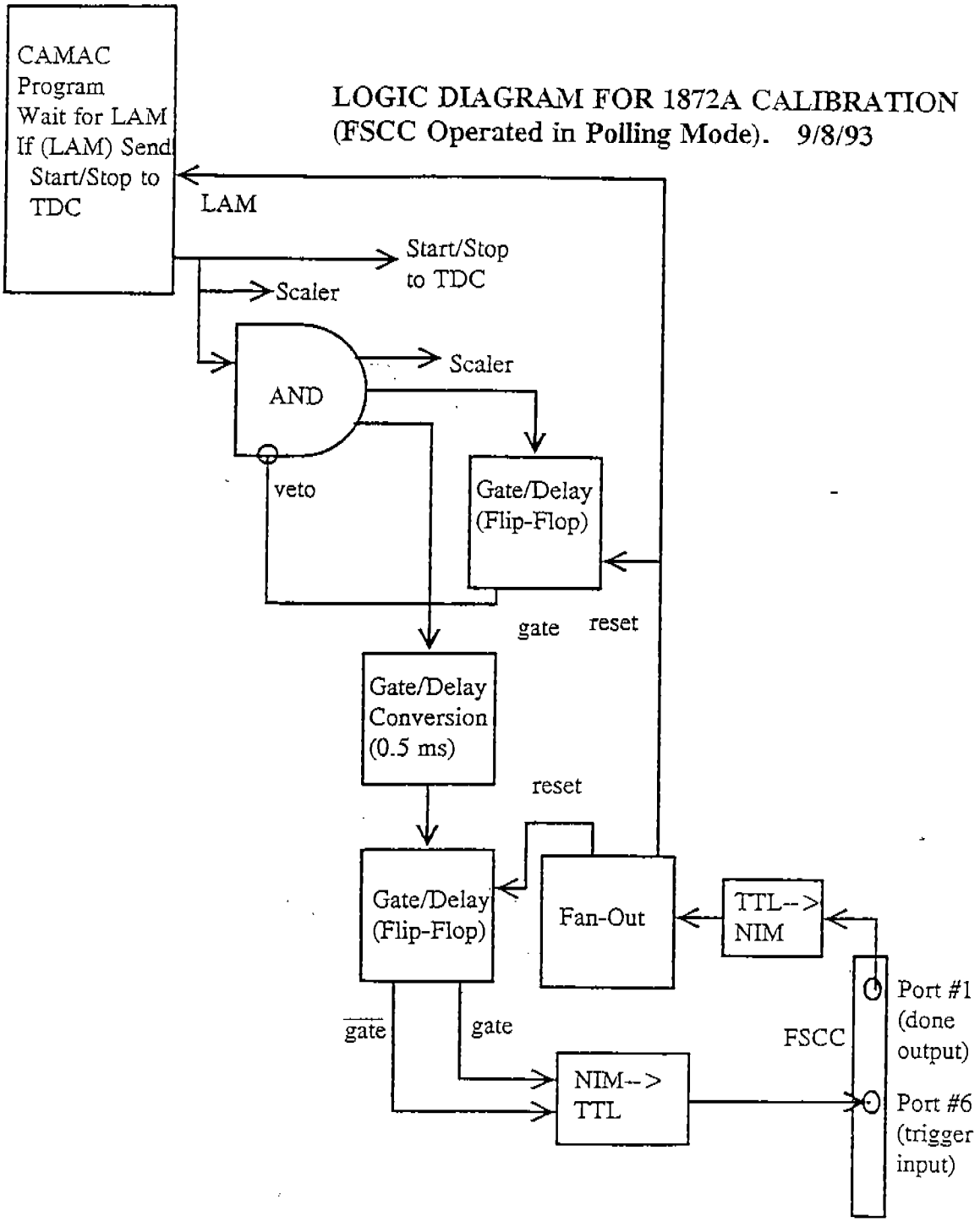

CAMAC
Program
Wait for LAM
If (LAM) Send
Start/Stop to
TDC
LOGIC DIAGRAM FOR 1872A CALIBRATION
(FSCC Operated in Polling Mode). 9/8/93
LAM
Start/Stop
to TDC
Scaler
AND
Scaler
veto
Gate/Delay
(Flip-Flop)
gate
reset
Gate/Delay
Conversion
(0.5 ms)
reset
Gate/Delay
(Flip-Flop)
Fan-Out
TTL-->
NIM
Port #1
(done
output)
FSCC
gate
gate
NIM-->
TTL
Port #6
(trigger
input)

Figure 3

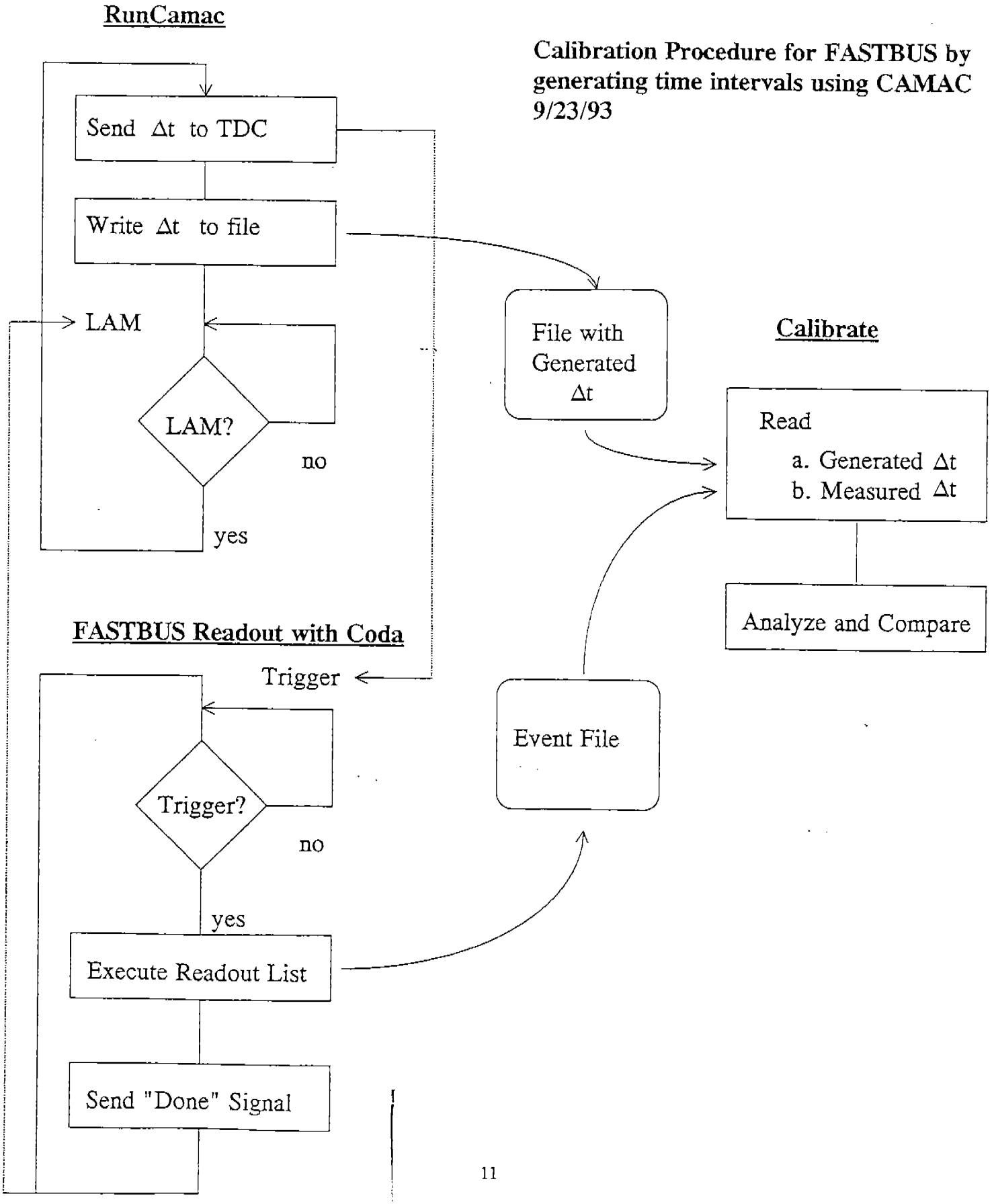

RunCamac
Calibration Procedure for FASTBUS by generating time intervals using CAMAC 9/23/93
Send Δt to TDC
Write Δt to file
LAM
LAM?
no
yes
File with Generated Δt
Calibrate
Read
a. Generated Δt
b. Measured Δt
Analyze and Compare
FASTBUS Readout with Coda
Trigger
Trigger?
no
yes
Execute Readout List
Send "Done" Signal
Event File

Figure 4

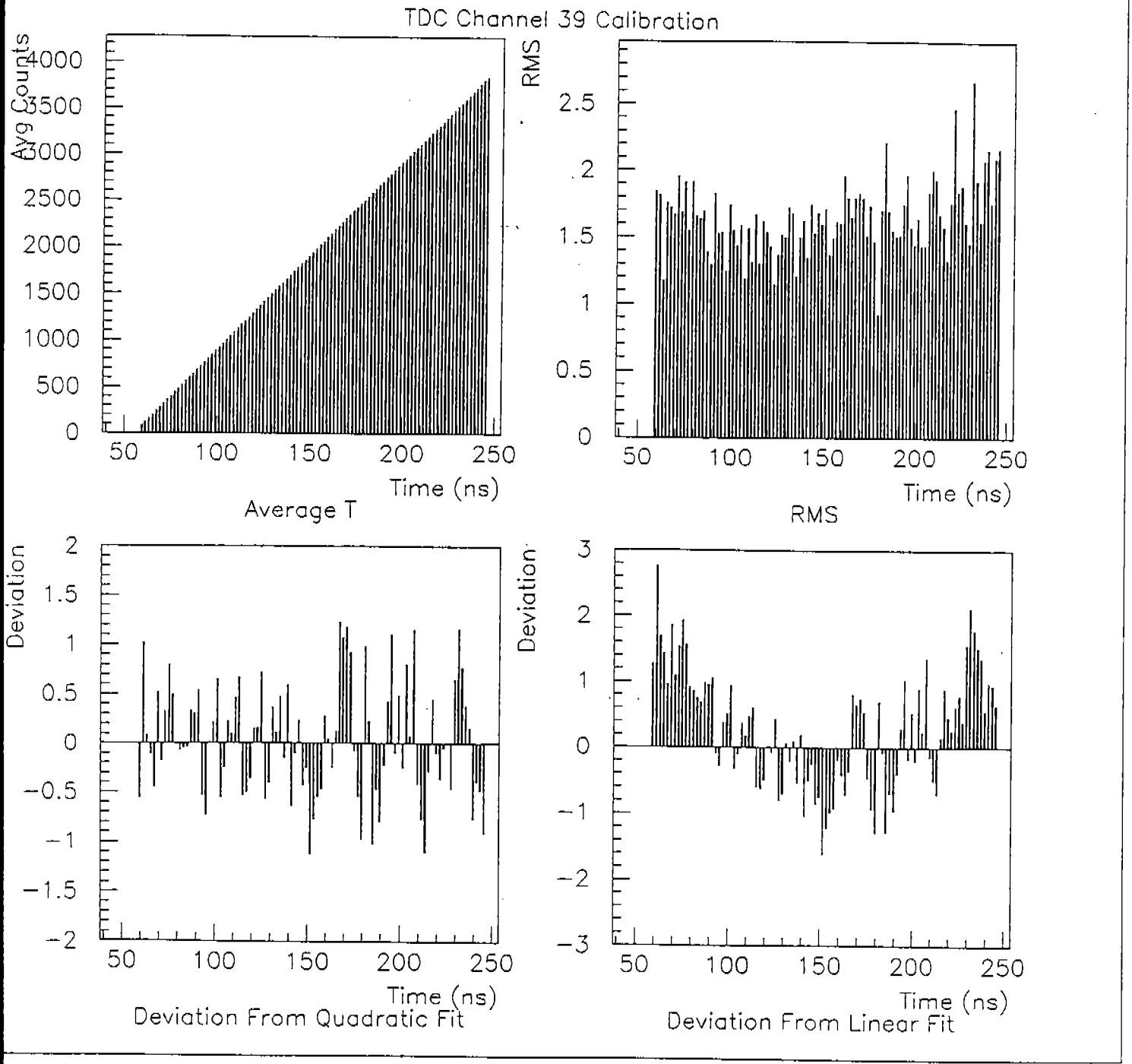
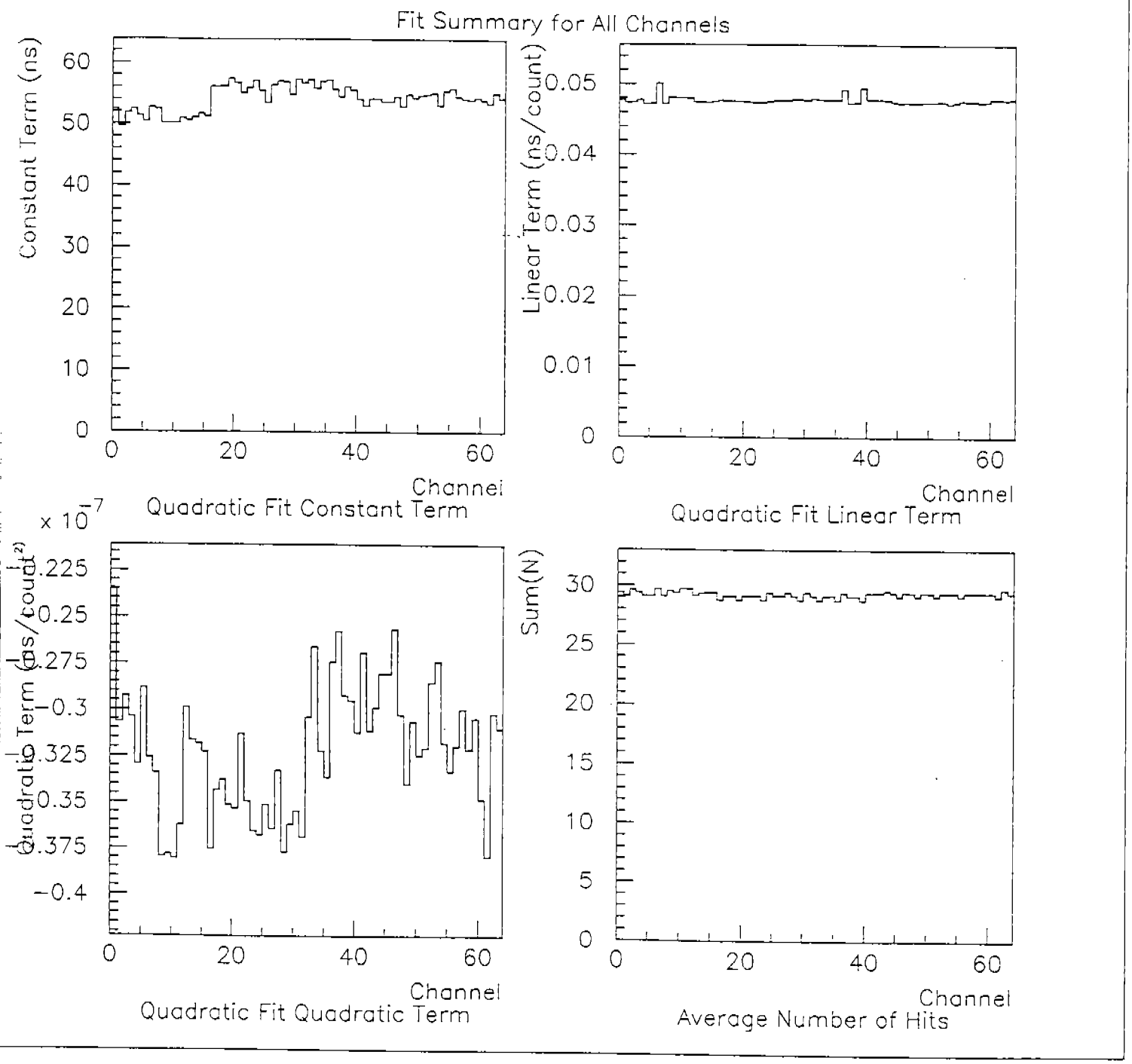

TDC Channel 39 Calibration
Avg Counts
4000
3500
3000
2500
2000
1500
1000
500
0
50
100
150
200
250
Time (ns)
Average T
RMS
2.5
2
1.5
1
0.5
0
50
100
150
200
250
Time (ns)
RMS
Deviation
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
50
100
150
200
250
Time (ns)
Deviation From Quadratic Fit
Deviation
3
2
1
0
−1
−2
−3
50
100
150
200
250
Time (ns)
Deviation From Linear Fit

Figure 5


Fit Summary for All Channels
Constant Term (ns)
Linear Term (ns/count)
Channel
Quadratic Fit Constant Term
Quadratic Fit Linear Term
Quadratic Term (ns/count²)
Sum(N)
Quadratic Fit Quadratic Term
Average Number of Hits

Appendices

RunCamac.exe.....A

Camac_Io.....B

Calibrate.exe.....C

JoinRun.exe.....D

Sample Coda Files.....E

rcNetwork.....E.1

lecroy1.config.....E.2

lecroy1.crl.....E.3

Appendix A

RunCamac.exe

(By Robert Millner 9/3/93)

RunCamac.exe is a program to send test data to the Phillips 7120 Charge Time Generator located in slot 7 on crate 0 of CLASVME1. The program allows you to specify a start, roof, floor, resolution, max hits, run type, delay interval, LAM access, and output file. The roof and floor define the high and low bounds of the Phillips 7120 Charge Time Generator (QTG). When the program passes one boundary, it will be set to the other. The start is the first value into the 16 bit DAC on the QTG. The resolution is the number of counts to skip on the DAC16 each step. Hits determine how many times to cycle through the complete range (hit each specified time) and run type (see the below table) details how the output data will be created. From that information, the number of events is computed and the program will cycle through all of those events. Delay interval is the amount of seconds to wait on each event. This parameter can be used to slow down the system in order to insure proper synchronization between CAMAC, FASTBUS and the data taking computer. LAM access will enable or disable the program's use of a LAM on a TDC in slot 10. The output file is a list of the counts that were sent each event. At the end of the run, information on the time, rate, number of events attempted and number of successful events is output to the terminal.

This program preforms two types of error checking. The first is a query to the CAMAC system preformed automatically by the Camac_Io subroutine. The returned word (in the variable pstat) is the status of the last operation. A successful operation returns a zero in this word. If it, or the status of the host is bad, then the event is re-delivered until it is successful. An error message is displayed along with the attempt number. The second type of error checking is preformed if the lam is enabled. If the program has not seen a LAM five seconds after delivery of the event (compensated for the user time delay) then the program will time-out and repeat the last event. The program will also add an internal delay of 0.02 sec prior to delivering each next event. This is increased 0.02 sec for each time-out that occurs. After ten errors or time-outs, the program will terminate.

This program was designed to be used in conjunction with fastbus TDCs and ADCs and Coda taking data on a host computer. The coda data file should only contain information on the FASTBUS TDC. The CAMAC information can be loaded in from the data file. From that, parameter fitting and calibration may proceed. Care must be taken to ensure that the synchronization is correct and that the 7120 triggers the process. There are possible bugs that will change the events and not trip the built in error checking. One way to ensure a proper setup is to do a test run sparsely in the full range with only a few hits. This data can be hand checked and run through the calibration program. Possible error signs include large RMS deviations (>2 counts) on all input times and poor quadratic fits.

The executable program, sources and the makefile are in /clas71/usr/local/coda/Calibrate. See the below table for available run types and the information below for upgrades.

Run Types

1..."Incremental Run": The data starts at the specified

start and cycles through the range max hits number of times adding the number of counts given by resolution each event.

2..."Random Run": The data is randomly chosen among slices of the range between the floor and the roof. The same number of events is delivered that would be done with run type 1 on a similar configuration. Resolution specifies how many points to skip each slice.

Code Modifications

Should it become necessary to modify the code, sources may be found in /clas71/usr/local/coda/Calibrate. Please comment and document what is changed. The first major section of the code is the opening menu. This starts by initializing the data to their defaults and then provides a user interface for changing the values. The defaults can be changed to match often used values in order to simplify using the program. New options may be added by including them in the init list, including a type statement, menu option and adding the proper check and service routines.

The second code section is the "prestart." This sets up parameters necessary to control the run. The location of the test and LAM modules can be changed by changing the variables c, n and a. The variable c is the crate number, n is the QTG slot number, nl is the LAM slot number, and a is the 16 bit DAC subaddress. The Camac_IO subroutine controls the branch number and MVME162 host name.

Run types may be added after line 21 in the if check. This section only gets called if the last event was successful. In order to add a new run type, insert an "elseif (CAMAC_RUNTYPE.eq.##) then" line and an action to be preformed on CAMAC_OUT. CAMAC_OUT is the variable that will be output to the DAC.

The next block of code prepares the LAM. To ensure that the next delivered event will trigger a LAM this block must be placed just prior to delivering the event. Immediately following this is the event delivery and error checking.

It may be necessary with different setups to preform the next event even if a LAM does not fire. If so, then comment out the lines that change the values of j and ok_update in the LAM error checking code. Disabling the internal delay may be done here as well. If a recompile is necessary, CODA, rpc, KB_tricks (From Kevin Beard), UTRIX_special (also from Kevin Beard) and Camac_IO (from Robert Millner) must be available. Questions or comments to Robert Millner (millner@cebaf.gov).

Appendix B

Camac_Io
(by Robert Millner 9/15/93)

Camac_Io is a subroutine for facilitating interaction between a user program and a Camac system through the Coda software. The object file and source can be found in /clas71/usr/local/coda/RunCamac. Additional libraries from coda need to be linked with this software. These are $CODA/lib/libca.a and /usr/site1/rpc/usr/lib/librpclib.a.

The calling conventions for Camac_Io are:

(void) Camac_Io(int crate, int slot, int address, int function, int *data, int *q, int *hstat, int *pstat)

or

integer*8 crate,slot,address,function,data,q,hstat,pstat

camac_io(crate,slot,address,function,data,hstat,pstat)

Crate.....The crate number on the camac interface (0 to 7)

slot Number.....Physical slot of the device being accessed (0 to 25)

sub Address.....Memory register being accessed (specific to the module)

Function code.....Valid CAMAC function code (0 to 32)

(see p.339 in the 1992 Lecroy Catalogue or camac.doc)

data.....A 24 bit integer to be read from or loaded to depending on function

q.....X and Q status (1= no Q, 2= no X, 3=neither)

hstat.....Host Status

(0=Host not active, 1=Host active, set not equal to 1 to open a new connection)

pstat.....Polled status (0=ok)

The subroutine will take care of establishing a connection to a host on the first attempt to use it. Currently, the subroutine makes a connection to CLASVME1. If it cannot do so, it will return -1 for both q and pstat in addition to returning status in hstat. The connection will stay open for the duration of the program, after which the operating system will close it. Since there are a finite number of connections, the program should only open one per run. The subroutine also does checking to see if the Inhibit on the CAMAC crate is set and clears it if so.

Changing the branch number or host id can be done at the beginning of the subroutine in the variable initialization. The source is in C, so "cc -c camac_io.c" should be used to produce the object module. The source code actually has two subroutines, the Camac_Io subroutine for C and a subroutine that links to FORTRAN. The fortran subroutine calls the other one with the proper variable conversions. Care must be taken in calling the subroutine from FORTRAN. Only actual variables may be passed or the program will stop on a segmentation fault.

Much credit is due to Chip Watson, CEBAF Data Acquisition Group for the cnaf program after which this subroutine was designed. Comments and questions to Robert Millner, or Millner@cebaf.gov

Appendix C

Calibrate.exe

(by Robert Millner 9/17/93)

The calibrate program is designed to be used to bring together the raw information and produce informative output in an experiment to test and calibrate one Lecroy 1872A TDC module in fastbus that is having test information sent to it by a Phillips 7120 Charge Time Generator (QTG) on camac. The program inputs a file of physics events and outputs a table of averages, deviations, fits and deviations of the input data from those fits. Sources, makefile, executable and docs are in /clas71/usr/local/coda/Calibrate.

Calibrate requires two input files, the Coda log file containing the fastbus information and the RunCamac output file. The program will also look for two other input files and operate with them if found. The first of which is, the offset file, is a table of real times to be subtracted from the input time. A different time is given for each channel and the offset is subtracted any time a real time is to be used in calculation. The second file is a file of old parameter fits. At the moment, the program does nothing with this file; however, access to it is left as an upgrade option. If any of the required files is missing or cannot be opened, the program will stop on error. If any of the nonessential files is missing, the program will ignore any operations dependant on that information.This program will produce three output files: a table of parameter fit coefficients, an HBOOK metafile and an rzdat file. At the time of this writing, the rzdat file is useless but was included in as an upgradable option.

The names of the previously mentioned input and output files are queried by the program in a text menu that is activated in the initialization. There are two options in addition to the filenames. The first of these is the Phillips 7120 Charge Time Generator linear fit parameters to convert counts into time. The second option is a True/False setting to use the HIGZ window to view the metafile as it is being made. This was included primarily as a debugging option. If it is set to false, then HIGZ will only produce a metafile. Using the screen output adds considerable time to the execution of the program and is not recommended. All menu options have specified defaults. The defaults are the same as the defaults for the RunCamac program and the example CODA files written for this experiment in order to simplify the use of this program.

When run, the program will initialize itself, open its files and begin to load in events. Events are decoded by the CoEvb_unpack library; therefore, the program will be able to decode any device that the library can. In addition, the program makes no distinction between different slots. Information is stored according to its channel number and the corresponding time in the camac output file. The data space and algorithms are optimized around a long duration of runs with some kind of repeating pattern. The size of the data space can be changed in the parameters file. This will be explained in the modifications section. At current capacity, the program can handle one-thousand different input times on sixty four different channels for thousands of cycles. If in the data taking cycle, the program runs out of places to put input times, it will warn the user and continue on, ignoring that event. The program will continue reading events until an end event is encountered. This is decided by the CoEvb_unpack library[F].

Information from a run is discarded for the following reasons. First, only what the unpack

library thinks is a valid device in a valid physics event is decoded. Second, if an event does not have fastbus information, the event will be ignored. Third, any channels that return values of zero or FFFh (the maximum output for the 12 bit ADC) will be ignored. Fourth, any channels outside of the range specified in the parameter list will be ignored. Finally, only input times in a specified range will be used for computing the parameter fit constants but all points will be used in calculating averages and means.

The output information is built after the data taking run. Since the program is producing a metafile in this segment, it will take some time to complete. A page of histograms will be produced for each channel that the program has usable information from. Any missing channels in the run will not produce output. Each page has four histograms, the first is a list of average counts vs. input time. The second histogram is a table of standard deviations from the averages. The third and fourth are plots of the deviations from the linear and quadratic fits. The x axis range is computed from the bounds of the input time. The range of the averages and deviations is calculated by the HBOOK routines. The output of the deviations from fit parameters is shown in the ranges of [2,-2] and [3,-3] for the quadratic and linear fits respectively. The output is designed to be used for qualitative analysis of the performance of the TDC. The fit parameters are histogramed at the end of the run for comparison and output to a file. There are two forms of parameter fits, TDC counts to time and TDC time to counts. Outputting either one or both can be done by changing the source code. At the time of writing, the fit produces time in nsec for TDC counts. The output fit constant file is a 1024 length record file of format "integer real real real real real." The integer is the channel number. The first three reals are the quadratic fit terms in the order of constant, linear and quadratic. The last two reals are the linear fit terms in the order of constant and linear term.

Known Bugs:

1. A segment fault occurs in closing the input file. This is a bug in one of the libraries. It is placed at the end of the program to avoid interfering with the rest of the run.
2. On metafile screen output, the buffer does not always clear to the screen. This is not a catastrophic error as the metafile is still produced properly.
3. If there are several devices in the log file, the program will overlay information form the channels. This will have strange results. The program was written with one (any) slot in mind but a check could be included in the GetIn_event if necessary to keep track of the slot number.
4. The program will get stuck in a read loop if the log file is missing a prestart, run or end event. The unpack library makes seeing and end dependant on seeing a start which is dependant on seeing a prestart. For normal data, this is not a problem; however, if this becomes a problem, the CoEvb_unpack library could be recompiled to remove this dependency check.

Parameter file: Calibrate.PAR

MAX_NUM_CHANNELS.....Number of input channels -1
OFFSET_NUM_CHANNELS.....Channel to start with in the run
MAX_DATA_LEN.....Maximum number of different input times

MIN_FIT_BOUND_TIME.....Lower boundary on the input time
MAX_FIT_BOUND_TIME....Upper boundary on the output time

Source Files:

Calibrate.for: This is the main body of the program. It calls the init procedure at the beginning of the run, preforms data taking cycles, does post analysis, and calls procedures to output the metafile.

Calibrate_inits.for: This is the initialization subroutine. It produces the menu and handles variable initialization and opening files. It also determines weather or not enough information is available to continue the program.

GetIn_Event.for: This subroutine uses the load sub to get an event and puts the information into user arrays. In addition, this subroutine screens out unwanted information from the file and determine weather or not an event has enough information to be useful.

load_physics_event.for: This subroutine takes care of using the evgen routines to open and close the events file and read in an event. It also calls the CoEvb_unpack subroutine to unpack usable information and decode the TDC data. It also takes care of opening and closing the camac file and produces an array containing the event.

Get_Position.for: This is a subroutine for allocating and searching places along the data space for storage of events. The subroutine employs a linear search to determine weather or not a time has been loaded before. If so, it returns the address of that time, if not, it allocates new record and initializes it. If it tries to allocate an event but there is no space then an error is returned. The subroutine remembers where the last event was placed and searches there first on the next event. It then searches to the end of allocated space, loops around from the beginning and searches until it reaches where it started or locates the target time. This was done to optimize filling in a small, sparse pattern which repeats often.

Re-compiling the program requires access to the evgen and evfio libraries by Larry Dennis, the KB_tricks and ULTRIX_special libraries by Kevin Beard, CoEvb_unpack by Robert Millner, CERNLIB and the CODA libraries.

Questions or comments to Robert Millner, millner@cebaf.gov

Appendix D

JoinRun.exe

(by Robert Millner, 9/17/93)

The join run utility will take up to a hundred CODA Raw Physics Event files and concatenate them into one run. This is useful for calibrating FASTBUS modules where only one part of one device can be accessed each run. Instead of passing around several files and operating on each one at a separately, this merges the files to allow them to be operated on together.

The CODA "prestart" and "go" events in the output file will be taken from the first input file. The "end" event comes from the last. The only other banks that are transferred are the physics event banks which are filled in the order of each file. This can be changed in the source code by modifying what banks are looked for. The program will query the user on the names and number of the input files and the name of the output file. If the output file already exists, the program will overwrite the file. The output filename must be different from the input filenames. After filling the file, the program will report on the number of physics events transferred and close the files.

The executable, source and makefile are located in /clas71/usr/local/coda/JoinRun. Access to the CODA libraries is necessary for recompilation of the program. The program uses the evOpen, evClose, evRead and evWrite subroutines to preform file IO. Descriptions of these subroutines can be found in the CODA manual, chapter 4, pages 9 to 11.

Questions or comments to Robert Millner, Millner@CEBAF.GOV.

Appendix E.1
"rcNetwork"

```
!+
! File:-
!         rcNetwork
!
! Description:-
!         Network Configuration file.
!
!         A Host of $NODE implies the same node as RunControl.
!-

!Name     Num  Type     Host     BootScript
!----     ---  ----     ----     ----------

FASTBUS  0    ROC      clasfb1
analyzer 0    ANA      $NODE     $CODA/bin/coda_activate -f $CODA/
bin/coda_ebana
```

Appendix E.2
"Lecroy1.options"

```
!+
! File:-
!         lecroy1.config
!
! Description:-
!         Lecroy1 RunType Configuration File. Read LeCroy 1872A

FASTBUS   $RCDATABASE/lecroy1.o
! EventB     LOG
analyzer  $RCDATABASE/Data/lecroy1.log
```

Appendix E.3

```
!
!    Stephen Wood. CEBAF August 1992.
!
!    List to readout the Lecroy 1872A TDC.
!
! Modifications by Robert Millner August, 1993

inline fastbus /*Optimization*/
fastbus readout /*Fastbus Code, Full Status Messages*/
polling  /*Less intrusive than interrupt mode*/

TDCLSLOT = 17        /*Location of the TDC*/

begin download
!  log warn "lecroy.crl ->Empty Download procedure"
end download

begin prestart

  variable temp

  address geographic control TDCLSLOT
  write hex 80000000   /*Digital CLEAR*/
  write hex 40000000   /*RESET*/
  ! Disable autorange, Set range low
  ! Set MPI Internal, Disable test pulser
  write hex 35000240
  read into temp
  log warn "lecroy1.crl->TDC CSR0 = H8", temp
  release

end prestart

begin end

!  log warn "lecroy1.crl->end list executing - trigger disabled"

end end

begin pause
```

```
!  log warn "lecroy1.crl->pause list executing - trigger disabled"

end pause

begin go

log inform "lecroy1.crl -> GO list executing"
address geographic control TDCLSLOT
write 0x80000000 /*DIGITAL CLEAR*/
release

end go

begin trigger

address geographic data TDCLSLOT
block read      /*Transfer all data.Automatically clears.*/
release

end trigger

begin done
end done

begin status
end status
```

Bibliography

A CEBAF SPECIFICATION 66340-S-00846, "Time to Digital Converters for TOF Measurements with CLAS"

B Phillips 7120 Manual. P 1.

C Phillips 7120 Manual. P 2.

D LeCroy 1992 Research Instrumentation Catalog. P 85.

E CLAS-NOTE-93-018, "TOF Temporary Software", by Robert Millner. 1993. P. 2

F CLAS-NOTE-93-018, P. 3

G LeCroy Catalogue, p. 41