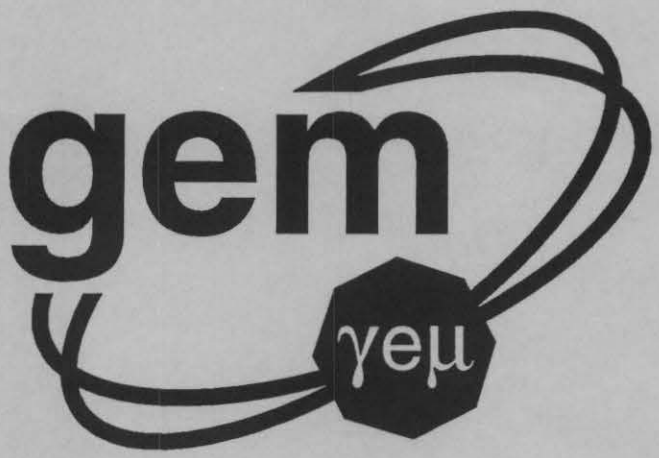
GEM TN-93-369



Texas Test Rig Data Acquisition System (UNIX / VxWorks Based System)

M. Botlo, M. J. Jagielski, A. Romero
SSC Laboratory

September 25, 1992

Abstract:

The objective of this document is to give an overview of the hardware and software portions of the Texas Test Rig (TTR) Data Acquisition (DAQ) system. This system utilizes the X window graphical user interface (GUI) of a Sun SPARCstation along with the real-time predictability of VxWorks running on a Motorola (MVME147) 68030. Also presented in this document is the performance of the TTR DAQ system.

TEXAS TEST RIG DATA ACQUISITION SYSTEM

(UNIX / VxWorks Based System)

Version:

25-September-1992

M. Botlo, M. J. Jagielski, A. Romero

Superconducting Super Collider Laboratory (SSCL)
Physics Research Divsion

Table of Contents
TTR DAQ SYSTEM

1 INTRODUCTION

The objective of this document is to give an overview of the hardware and software portions of the Texas Test Rig (TTR) Data Acquisition (DAQ) system. This system utilizes the X window graphical user interface (GUI) of a Sun SPARCstation along with the real-time predictability of VxWorks running on a Motorola (MVME147) 68030. Also presented in this document is the performance of the TTR DAQ system.

Two data links exist between the SPARCstation and the 68030, a command / response link and an event data link. The TTR DAQ system can be configured to use the MXIbus and/or Ethernet for both of the links. The default configuration is to use MXI for the event data link (fast point-to-point connection between VME and the data logger) and Ethernet for the command / response data link (standard but slow).

Chapter 2 and 3 cover the hardware and software configurations respectively, while chapter 4 covers the perfomance of the TTR DAQ system.

Appendix A contains the TTR DAQ System User's Guide, covering from start up, through configuration of a read out, to monitoring a read out.

The following terminology will be used throughout the rest of this document:

GUI system	Sun SPARCstation
real-time system	Motoroloa 68030
process	executable running under UNIX on the SPARCstation
task	executable running under VxWorks on the 68030

2 HARDWARE CONFIGURATION

This section describes the TTR DAQ hardware configuration. As mentioned in the introduction, there exist two data links, each independent from the other where each can use MXI or Ethernet as a means of transferring data. Figure 2.1 shows the hardware configuration for the TTR DAQ system. The following sections give a brief description of each component and its function in the TTR DAQ system.

2.1 CAMAC

In the TTR DAQ system, the Dsp Technology CAMAC crate contains a GPIB Crate Controller, a CAMAC trigger module and front end modules for data conversion and for data read out.

2.1.1 GPIB Crate Controller

The KineticSystems 3988 GPIB Crate Controller allows system controllers to communicate with up to 23 CAMAC I/O modules in a crate. The model 3988 supports block transfers at data rates up to 600 kilobytes per second with the actual rate limited only by the controller and/or the host computer. Usually Q-Scan operating mode is used to transfer a block of data between a group of modules and the GPIB host computer (MVME147).

2.1.2 NIM/CAMAC Interrupt Module

The NIM/CAMAC Interrupt Module TU7800 generates CAMAC Look-At-Me (LAM) and NIM Bin Gates on receipt of NIM logic pulse. NIM Gates can be enabled or disabled.

2.1.3 ADC, TDC and other Modules

Each LeCroy 2249A ADC (Analog-to-Digital Converter) has 12 channels. The module digitizes all 12 channels in parallel to 10 bits each in 60 μs. Once complete a LAM can be generated.

The Phillips Scientific 7186H TDC contains 16 channels for Time to Amplitude Conversion (TAC).

2.2 GPIB

The National Instruments GPIB-1014 is a bus translator, converting messages and signals present on the VMEbus into appropriate GPIB messages and signals. The device has a DMA controller for data

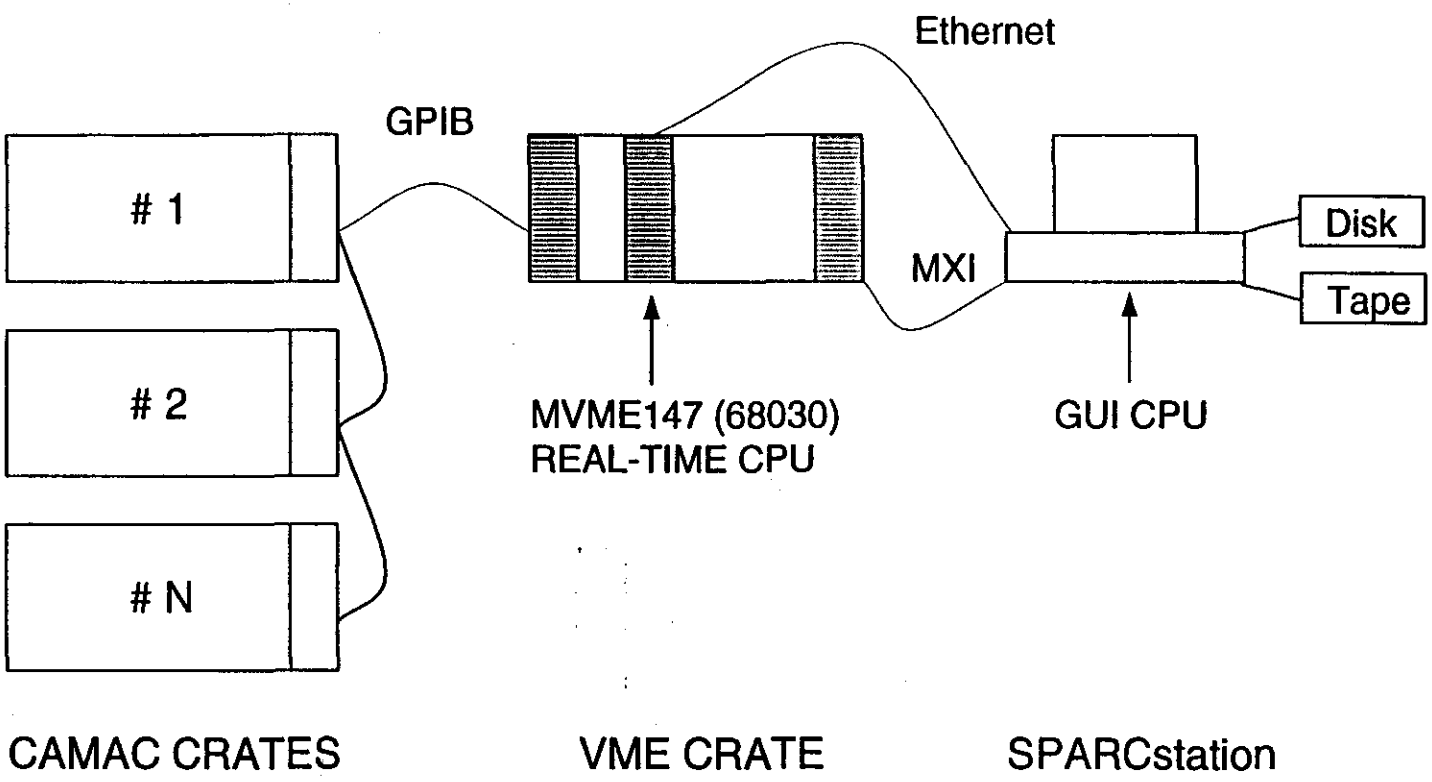


Figure 2.1 TTR DAQ HARDWARE CONFIGURATION

transfers to and from the GPIB.

The following are typical restrictions:

- A maximum separation of 4 m between any two devices and an average of 2 m over the entire bus.

- A maximum total cable length of 20 m.

- No more than 15 devices connected to each bus, with at least two-thirds powered on.

2.3 MVME147

The MVME147 has the MC68030 microprocessor with a 32 MHz clock and 8 MB of RAM. The MVME147 also has a floating-point coprocessor, Ethernet transceiver, and small computer system interface (SCSI) bus.

2.4 SPARC

The SPARCstation 2 is a Sun Microsystems workstation running SunOS release 4.1.2 and OpenWindow 3.0 graphical windowing environment. The workstation comes with 32 MB RAM and 424 MB internal SCSI disk. It also can get a network connection through its built-in ethernet port. The 19 inch color monitor is driven by a GX graphic card which can provide a pleasing graphical user interface and amazing three-dimensional graphic displays.

2.5 MXI

The MXIbus, Multisystem Extension Interface bus, is a multi-drop parallel bus architecture designed for high speed comunication between devices.The MXIbus architecture provides a very high performance link between two devices because it maps actual bus cycles on one device to bus cycles on another device (see MXIbus reference, PR/DAQ note, A.Romero et. el).

In the TTR DAQ system a National Instruments (NI) VME-MXI module is installed in a slot of the VMEbus chassis. This module interfaces the VMEbus to the MXIbus. NI also provides the SB-MXI module which links the Sun SPARCstation (using the Sbus expansin slots) to the MXIbus.

Altogether these interfaces allow data to travel from VME memory through the MXIbus to the Sun SPARCstation. From there the data can be stored on disk or tape.

2.6 Ethernet

Ethernet can be used as the link between VME and the SPARCstation for both the command / response data link and the event data link. As will be presented in chapter 4, ethernet will give performance degradation when used on the event data link.

2.7 EXB-8500

Data recording will take place using the EXABYTE Corporation EXB-8500 8mm Cartridge Tape Subsystem. The EXB-8500 is a high performance, high capacity 8mm cartridge tape subsystem. The EXB-8500 can achieve a transfer rate of up to 500 KBytes per second. The EXB-8500 will be connected to the SPARC workstation using the SCSI controller. This device can be used in 8200 or 8500 mode using different user selectable drivers.

3 SOFTWARE CONFIGURATION

This section describes the TTR DAQ application software. This software is managed in two segments; the software which runs in the GUI system and the software which executes in the real-time system. Figure 3.1 shows the default software configuration for the TTR DAQ system with MXI being used for the event data link.

3.1 Graphical User Interface (GUI) System

The TTR DAQ system takes advantage of the OPENLOOK GUI application environment provided by the Sun SPARCstation to give the user a efficient way to control the system. The main GUI process manages the window environment and selections the user is making. Three additional processes execute on the SPARCstation as well (See Figure 3.2). The first, TTR Server, provides the communication path between all other processes and the real-time environment on the 68030. The second, Data Collection, gathers the data from the 68030 and stores it on hard disk, while the third, Tape Control, is a tape deamon streaming off data to cartridge tape and also creating a zero suppressed file.

3.1.1 TTR GUI

The TTR graphical user interface (GUI) process uses various sub windows to configure, control and monitor the data acquistion system. The user can observe real-time and system alarms along with other status on the main control panel. The following sub windows can be used: crate editor, tape setup, event dump, comment event, and show data display (See Figure 3.3). One feature of the ttrgui is that multiple gui's can be executing by multiple users at the same time. These gui's communicate through the interprocess communication mechanism of shared memory. Therefore all of the processes must be executing on the same machine. Appendix A explains how a gui can be displayed on the user's host machine. Also since multiple gui's could cause read out conflicts, only one gui will be allowed to be the master (actively interact with the DAQ system) while all other gui's are slaves only able to observe the read out. This is explained in more detail in Appendix A.

3.1.2 TTR Server Process

The TTR Server process handles all communications between processes on the GUI system using the interprocess communication mechanism of message queues. This process also uses ethernet to communicate messages between the real-time system and other GUI system processes. Each process on the GUI system registers with the Server process and thus all are notified if there is a fatal error within the data acquistion system and can shut down.

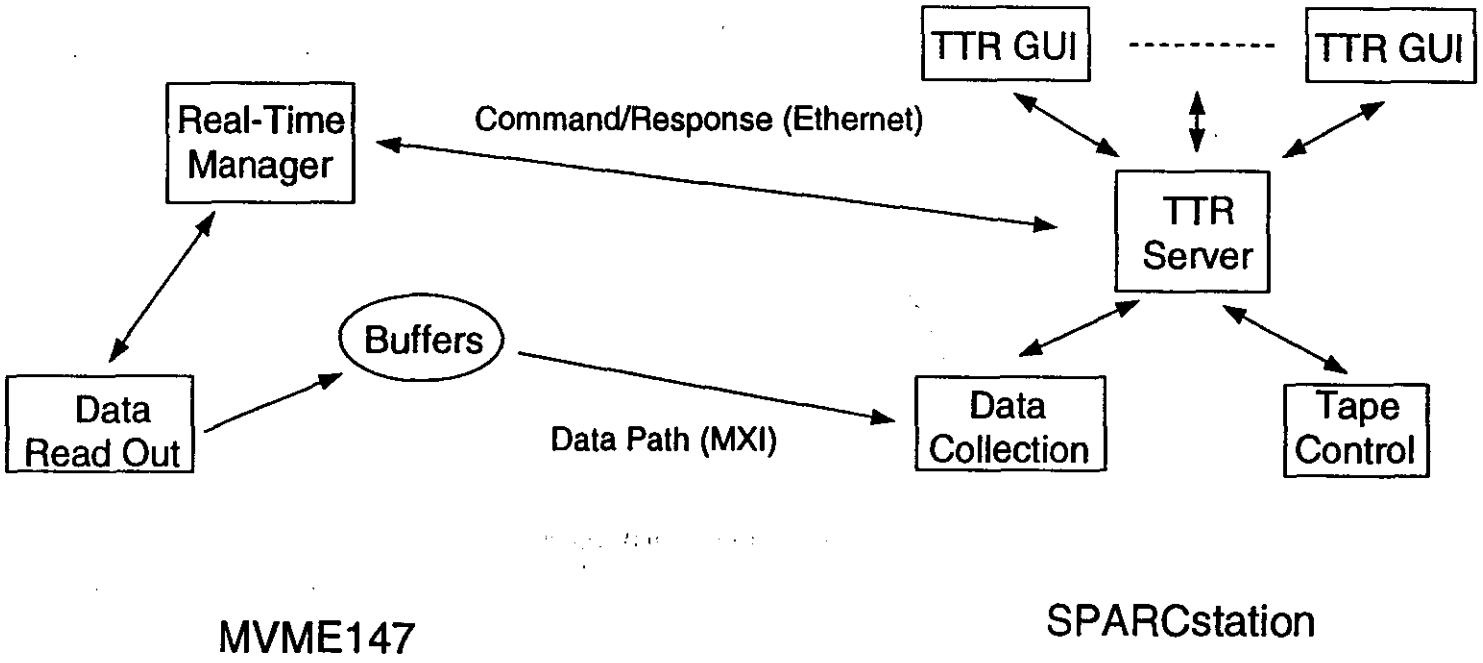


Figure 3.1 TTR DAQ SOFTWARE CONFIGURATION

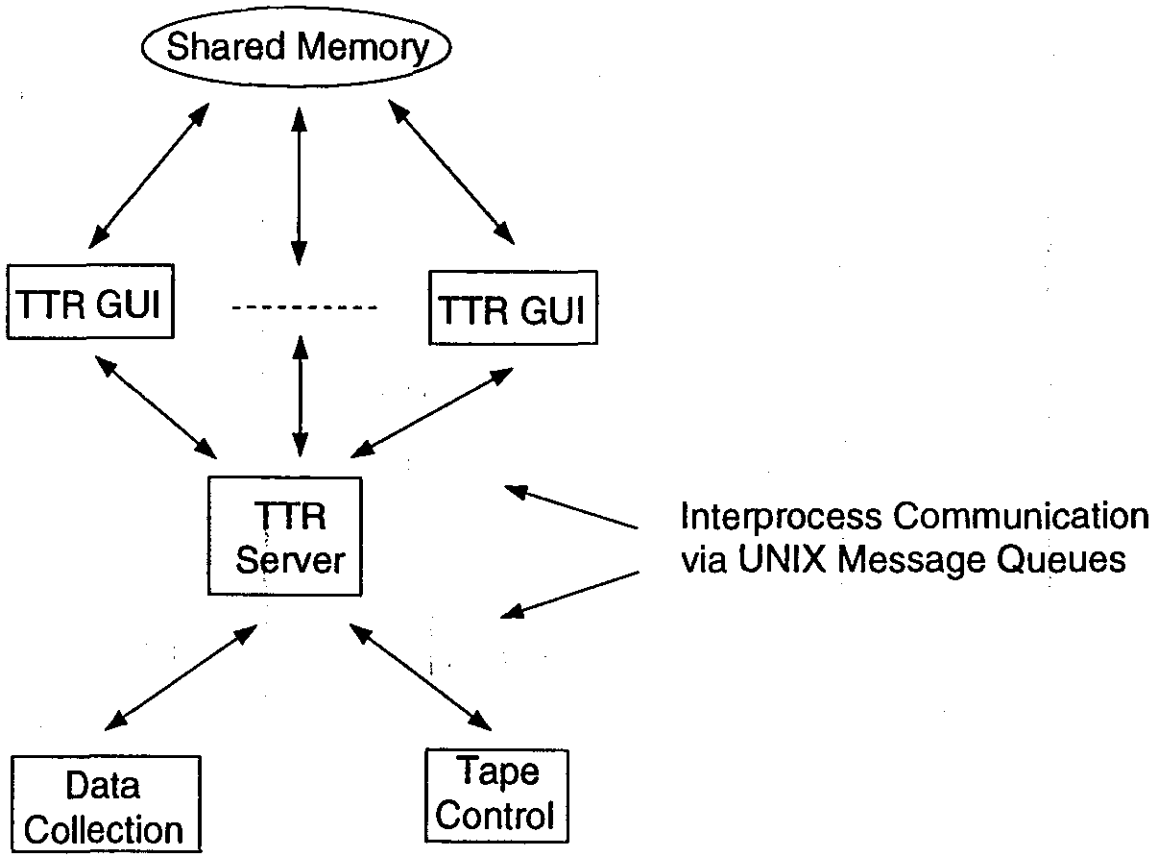


Figure 3.2 TTR SPARCstation S/W CONFIGURATION

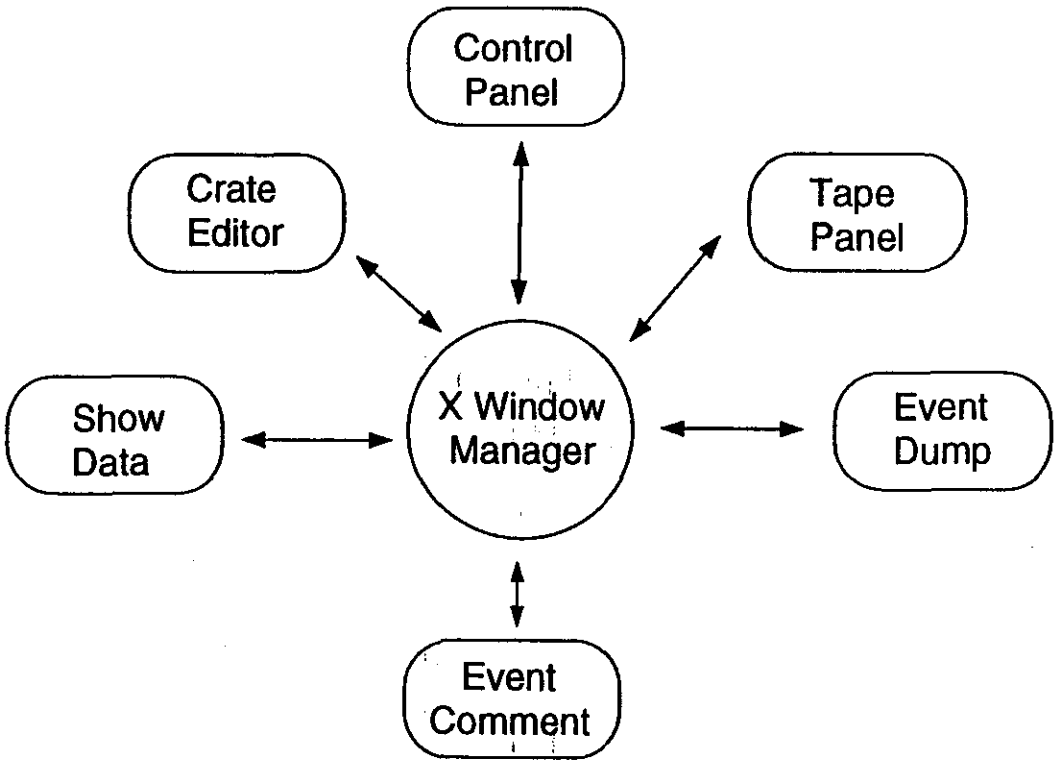


Figure 3.3 TTR GRAPHICAL USER INTERFACE (ttrgui)

3.1.3 Data Collection Process

The Data Collection process handles gathering all event data from the real-time system. This process checks to see which rings of data are full and reads them across the MXIbus. Once this is complete the process releases the rings for the real-time system to reuse. Once the data is received by the process it is written to disk. The disk acts effectively as a large and fast storage buffer for data. Once a pre-determined file size , user selectable, is reached this file is closed and passed to the Tape Control process.

3.1.4 Tape Control Process

The Tape Control process remains idle until it is passed a file name to be written to tape. Once the file is written to tape the file is destroyed or saved based on a set flag and the write information is logged in the tape database. If the zero suppression switch is on, then this process will also create a zero suppressed file on hard disk.

3.2 Real-Time Data Aquisition

At the heart of the real-time software is VxWorks 5.0.2. The real-time system operates on the 68030, giving the TTR DAQ system multitasking with preemptive priority scheduling, intertask synchronization and communication, interrupt handling support, timers, and memory management. The next three sections describe the three real-time application tasks running in the TTR system.

3.2.1 TTR Executive Task

The TTR Executive task is the first task to be spawned off when the TTR system is started. This task handles all initialization and is the main link between the SPARCstation using a command/response mechanism for communication. This task spawns off the Read Out task and then does a handshake with the SPARCstation using the command/response buffers. Once all initialization is over this task simply waits for a command from the SPARCstation. The commands it will respond to are: start, pause, stop, quit, probe, event dump request, insert comment, and status request.

3.2.2 Read Out Task

The Read Out task is truly an event driven task, as it waits for a service request interrupt from the printer port. This port is connected to whatever device is generating the external triggers. Once this is received, using the configuration specified by the SPARCstation, the task reads out data from the CAMAC crates and formats the event structure. The data is placed into a ring of buffers. Once a ring is full of data the SPARCstation (MXI) must release the buffer for reuse.

3.3 TTR Event Format

This section describes the TTR event format and provides details for expected values in each field.

3.3.1 Event Format Structure

The event format structure for the TTR DAQ system is as follows:

HEADER	Event Size	U16
	DAQ Mask	U16
	Trigger Mask	U16
	Run Number	U16
	Event Number	U32
	Date and Time	U32
GROUP 1	Group Size	U16
	Group Mask	U8
	Technology Type	U8
	Crate Number	U8
	Start Module	U8
	Number Of Modules	U8
	Module Type	U8
	Group Data	
GROUP 2	Group Size	U16
	Group Mask	U8
	Technology Type	U8
	Crate Number	U8
	Start Module	U8
	Number Of Modules	U8
	Module Type	U8
	Group Data	
	.....	
GROUP N	Group Size	U16

	Group Mask	U8
	Technology Type	U8
	Crate Number	U8
	Start Module	U8
	Number Of Modules	U8
	Module Type	U8
	Group Data	
END OF EVENT	0xFFFFFFFF	U32

In some cases, explained more below, the groups are simple replaced with data as follows:

HEADER	Event Size	U16
	DAQ Mask	U16
	Trigger Mask	U16
	Run Number	U16
	Event Number	U32
	Date and Time	U32
BODY	Data (Usually ACSII comment) ...	
END OF EVENT	0xFFFFFFFF	U32

3.3.2 Event Header

The event size is the number of 16-bit words from the beginning of the event, including itself, to the last byte of the END OF EVENT trailer. This convention should facilitate skipping events. All events will be 4-byte (long word) aligned, hence the event size should always be even. To ensure this, all group data must be 4-byte aligned. Extra zeros will be appended to the end of group data if this be the case.

The DAQ mask provides the type of event as well as whether the data is zero suppressed or not, encoded in the following way:

Bits 0-3	0 - Normal Event
	1 - Comment Event
	2 - Lower Threshold Event
	3 - Upper Threshold Event
	4 - Special Event
	5 - Initial Comment Event

	6 - Voltage Comment Event	
	7 - Calibration Comment Event	
	8 - Monte Carlo Generated Event	
	9 - 15 For Future Use	
Bit 4	0 - S/W Zero Supp. Off	1 S/W Zero Supp. On
Bit 5	0 - Macintosh	1 UNIX
Bits 6-15	For Future Use	

The first four bits of the DAQ mask define the type of event. The normal, lower and upper threshold, and special events all use the first event structure as defined above with event header, groups and end of event. The remaining event types are "comment" events which instead of having the group structure between the event header and end of event have ACSII characters as described above. These characters will always be 4-byte aligned just as the group data is in all other events. If extra bytes are needed zeros will be appended to the end of the data.

Software zero suppression (bit 4 of DAQ mask) will only be executed on specific groups within normal events. All other event types will be passed on touched. The module type will determine which groups have been zero suppressed.

Bit 5 of the DAQ mask will be used to indicate what system dif the read out whether it be the MAC or the UNIX based system.

The Trigger Mask will encode the type of trigger (we will be able to trigger on specific hit patterns of the scintillation counters, and we may also have L2 trigger based on hit patterns on the Iarocci tubes) as well as bits to indicate calibration runs. The trigger mask can be encoded in the following way:

0 - All Angle Muon Trigger (Level 2)
1 - Vertical Muon Trigger with X-Displacement (Level 2)
2 - Vertical Muon Trigger (Level 2)

The run number will be obtained by automatically retrieving and incrementing by one the previous run number from a file on hard disk. The event number is only reset at the start of a new run.

The date and time will be given as the number of seconds expired since 12:00 a.m., January 1, 1904. (for MAC - DAQ compatibility)

3.3.3 Group Header

The group size is the number of 16-bit words from the beginning of the group, including itself, to the last byte of data. This convention should facilitate skipping groups. Again this value will also always

be even as the group itself will be 4-byte aligned.

The second field in the group header is the technology that this group belongs to and can be encoded as follows:

0 - No Technology
1 - LSDT
2 - CSC
3 - PDT (Russian)
4 - PDT (USA)
5 - RPC
6 - Honeycomb
7 - CSC (Houston)
11 - Magnet
12 - Scintillators
13 - Iarocci's

The crate number specifies the CAMAC crate address the 3988 GPIB controller is set to. In general, the upper 3 bits specify whether CAMAC (000), VME (001), VXI (002), or any other front end buses are read out.

The next two fields provide the start module (slot) for the group and the number of modules contained in the group.

The last field in the group header describes the module type and can be encoded as follows:

0 - Empty Station (slot)
1 - LeCroy 2249A ADC
2 - Phillips 7186H TDC
3 - NIM TU7800 Trigger
4 - LeCroy 4434 Scaler
5 - Kinetic 3512 ADC
6 - LeCroy 2365 PLM
7 - LeCroy 4508 PLU
8 - STAS
9 - LeCroy 2277 TDC
99 - GPIB 3988 Controller

3.3.4 Miscellaneous Event Format Information

The following are TTR format numbering conventions:

Event Number	1 - N
Run Number	1 - N
Module Number	1 - 23

3.3.4.1 Other Formats

As was stated above the first four bits of the DAQ mask define the type of event. The lower and upper threshold events (along with the normal and special event) follow the standard event format with group headers and data embedded in the body of the event. In this case the data represents the threshold values.

The remaining events use the second format described in section 3.3.1 where the body of the event is a comment. The follwoing sections provide the format (or structure) for the "body" of these event types.

3.3.4.1.1. Comment Event

The format for the body of the comment event is as follows:

char string[2 * event size - sizeof(event header) - sizeof(end of event)];

3.3.4.1.2 Calibration Comment Event

The format for the body of the calibration comment event is as follows:

Event_Size	U16
DAQ_Mask	U16
Trigger_Mask	U16
Run_Number	U16
Event_Number	U32
Date_and_Time	U32
Group_Size	U16
Group_Mask	U8
Technology_Type	U8
Crate_Number	U8
Start_Module	U8

Number_of_Modules	U8
Module_Type	U8
A1	Real*8
B1	Real*8
A2	Real*8
B2	Real*8
....	
A64	Real*8
B64	Real*8

The constants A and B are the conversion constants from the TDC counts to time, in nanoseconds, for each of the 64 TDC channels used for our scintillators, ie.,

Time measured by channel i = Ai * Ni + Bi

where Ni is the TDC counts of channel i.

3.3.4.1.3 Voltage Comment Event

The user will have the decode the large ACSII string. The first line will be the command that was actually given to the high voltrage device. From here the user can determine the number of channels and slots. Here is an example of the voltage comment event:

read (0-1, 0-15)

Channel	Demand	Voltage	Current
(0, 0)	-2120	-2120.00	
(0, 1)	-2090	-2090.25	
(0, 2)	-1980	-1980.62	
(0, 3)	-2000	-2000.25	
(0, 4)	-2160	-2160.25	
(0, 5)	-2160	-2160.12	
(0, 6)	- 0	- 58.75	
(0, 7)	- 0	- 52.75	
(0, 8)	-2130	-2130.12	
(0, 9)	-2030	-2029.75	
(0,10)	-2090	-2089.87	
(0,11)	-2270	-2268.87	
(0,12)	-2180	-2180.12	
(0,13)	-1960	-1960.00	
(0,14)	- 0	- 56.00	

(0,15) - 0 - 2.62
(1, 0) -2020 -2020.87
(1, 1) -2030 -2030.50
(1, 2) -2050 -2051.12
(1, 3) -1990 -1990.62
(1, 4) -2130 -2130.37
(1, 5) -2120 -2120.87
(1, 6) - 0 - 38.50
(1, 7) - 0 - 57.75
(1, 8) -2150 -2150.75
(1, 9) -2080 -2080.75
(1,10) -2120 -2121.00
(1,11) -2180 -2180.87
(1,12) -2150 -2151.00
(1,13) -2010 -2011.00
(1,14) - 0 - 42.87
(1,15) - 0 - 2.12

0>

3.3.4.1.4 Initial Comment Event

The format for the body of the initial comment event is as follows:

Name of Shift Person	U8*32
Comment	U8*80
Number of Technologies	U32
(depending on the number of technologies, the following group may appear multiple times)	
Technology Name	U8*20
DAQ Usage	U32
High Voltage	U32 (Real*4)
Number of Gas Types	U32
(depending on the number of gas types, the following group may appear multiple times)	
Gas Name	U8*20
Percentage	U32 (Real*4)

4 TTR DAQ SYSTEM TIME MEASUREMENTS

This section provides detailed time measurements for the TTR DAQ system and helps the reader understand where the bottle-necks in the system exist. The end result is to reliably predict the event time given any DAQ configuration.

4.1 Test Run Conditions

In order to measure the read out time of the DAQ a 10 kHz trigger is generated to drive the acquisition. Event data is stored on the VME system in ring buffers, with 200 rings of 16384 bytes each being used for the test. When ethernet was used, the packet size was 16384 bytes. The system clock in VxWorks is set at 100 Hz and the auxilary clock is set at 400 Hz. The 8 registers available on the 68030 are used extensively in the read out task to maximize data buffering as much as possible.

4.2 Event Time Definition

The event time (ET) can be broken down into the following equation assuming that the number of modules per group (MODULE) and groups per crate (GROUP) are constant:

$$ET(C, G, M) = EO + CRATE * [CO + GROUP * (GO + MODULE * MO)] \qquad \text{Eqn. 4.1}$$

where

EO = RTEO + SETBG + WAIT + CLRLAM
- RTEO : Real-time event overhead
- SETBG : Set Bin Gate
- WAIT : Wait for SRQ
- CLRLAM : Clear trigger module LAM

CO = RTCO + READREQ + K * TCR + CLRCRATE
- RTCO : Real-time crate overhead
- READREQ : Request to read crate
- TCR : Initialize Transfer Count Register
- K : Percentage of TCR usage per event (depends on G, M)
- CLRCRATE : Clear crate (Only if TDC present)

GO = RTGO

RTGO : Real-time group overhead

MO = RTMO + CHAN * READ

RTMO : Real-time module overhead

CHAN : Number of channels in module

READ : Time to read a channel

CRATE : Number of crates

GROUP : Number of groups / crate

MODULE : Number of modules / group

Hence event time (ET), is a function of CRATE, GROUP, and MODULE. One can easily come up with derivative equations for when GROUP and MODULE are not the same for each crate.

4.3 Barebones Real-time Loop

Equation 4.1 above reduces to the following when no GPIB commands are considered:

ET(CRATE, GROUP, MODULE) =

RTEO + CRATE * [RTCO + GROUP * (RTGO + MODULE * RTMO)] Eqn. 4.2

The following time measurements were taken for various configurations:

(1) ET(1, 1, 8) = 86.9 µs
(2) ET(1, 1, 16) = 129.6 µs
(3) ET(1, 8, 1) = 184.3 µs
(4) ET(3, 1, 1) = 95.4 µs

Solving (1) and (2) yields RTMO = 5.3 µs.
Solving (1) and (3) yields RTGO = 13.9 µs.
Solving (3) and (4) yields RTCO = 3.6 µs and RTEO = 27.1 µs.

From these results one could predict what the result of ET(3, 1, 23) would be.

ET(3, 1, 23) = 27.1 + 3 * [3.6 + 1 * (13.9 + 23 * 5.3)] = 445.3 µs

The measured result was 442.7 µs which is well within the error constraints of the measurements.

NOTE: MXI does not influence the above results.

4.4 Barebones Real-time Loop with Ethernet

The same test runs as in section 4.3 were preformed this time adding the overhead of shipping the event data to the Sparc station via ethernet.

(1) ET(1, 1, 8) = 469.7 μs	(220 bytes / event)
(2) ET(1, 1, 16) = 846.3 μs	(412 bytes / event)
(3) ET(1, 8, 1) = 673.6 μs	(276 bytes / event)
(4) ET(3, 1, 1) = 293.3 μs	(116 bytes / event)

Using the result from section 4.3, the ethernet overhead for (1) is 382.8 μs (469.7 - 86.9) which results in 1.74 μs per byte (575000 bytes / second). The remaining runs (2), (3), and (4) result in the same ethernet overhead of 1.74 μs per byte. Again one could predict the result of ET(3, 1, 23), which would have 1700 bytes / event.

ET(3, 1, 23) = 445.3 + 1.74 * 1700 = 3403.3 μs

The measured result was 3422.7 μs.

4.5 GPIB Measurements

This section attempts to measure the times defined in equation 4.1 above that are a result of accessing the GPIB. Reducing the TCR update overhead will be discussed first. Then the smallest component, reading out a channel, will be examined along with solving the remaining unmeasured events.

4.5.1 TCR Update Overhead

The transfer count register (TCR) in the 3988 GPIB crate controller is a 16 bit register which decrements upon every 2-byte transfer of data. Depending on the number of channels being read out each event the TCR needs to be updated every N events. Initializing the TCR has been measured at 0.991 ms. Therefore this update does not take place every event as this would cause an unnecessary overhead. An internal TCR counter is maintained and updated every event in the read out task. For example if 264 channels (a channel is equivalent to a 2-byte transfer) are to be read out each event, then the TCR needs to be initialized every 248 (65535 / 264) events. This would result in a TCR update overhead of 4 μs (991 / 248) per event rather than 991 μs.

4.5.2 Time to Read a Channel

To examine the time it takes to read a single channel using GPIB, 1 crate was used with an initial group of 3 ADC's and another group of 5 TDC's. Each group was examined independently by decrementing a module for each run. Hence the delta time should be the time to read a module. First

the TDC module was examined:

	ADC's	TDC's	Total Time
(1)	3	5	5.881 ms
(2)	3	4	5.776 ms
(3)	3	3	5.670 ms
(4)	3	2	5.568 ms
(5)	3	1	5.463 ms

The average difference in each case is about 105 μs. Therefore one could solve MO = RTMO + CHAN * READ to determine READ = 6.25 μs (105 = 5 + 16 * READ).

Second the ADC module was examined:

	ADC's	TDC's	Total Time
(1)	3	5	5.881 ms
(2)	2	5	5.800 ms
(3)	1	5	5.720 ms

The average difference in each case is about 80 μs. Again one could solve for READ = 6.25 μs.

4.5.3 Other GPIB Functions

The three functions SETBG, CLRLAM, and CLRCRATE all use the common GPIB command dvwrt(). These functions were all measured in an isolated test and all resulted in the same time of 991 μs. The remaining unknowns in equation 4.1, WAIT and READREQ, can be solved for using the following two test runs:

(1)	Crate 1	Group 1	2 TDC's
		Group 2	1 TDC
	Event Size	132 bytes	
	Measured ET	5.463 ms	
(2)	Crate 1	Group 1	2 TDC's
		Group 2	1 TDC
	Crate 2	Group 3	3 ADC's
	Event Size	212 bytes	
	Measured ET	7.707 ms	

Note: Since crate 2 contains only ADC's there is no need of a CLRCRATE after every event.

These two runs result in the following equations:

$$5463 = 27.1 + 991 + WAIT + 991 + 3.6 + READREQ + 1.5 + 991 + 13.9 + 2 * 105 + 13.9 + 105$$

$$7707 = 27.1 + 991 + WAIT + 991 + 3.6 + READREQ + 1.5 + 991 + 13.9 + 2 * 105 + 13.9 + 105 + 3.6 + READREQ + 0.5 + 13.9 + 3 * 80$$

which reduce to

READREQ = 1986 μs
WAIT = 130 μs

Finally, a third run was made to check our results.

(3)	Crate 1	Group 1	2 TDC's
		Group 2	1 TDC
	Crate 2	Group 3	3 ADC's
	Crate 3	Group 4	2 TDC's
	Event Size	284 bytes	
	Measured ET	10.919 ms	

$$\text{Predicted ET} = 27.1 + 991 + 130 + 991 + 3.6 + 1986 + 1.5 + 991 + 13.9 + 2 * 105 + 13.9 + 105 + 3.6 + 1986 + 0.5 + 13.9 + 3 * 80 + 3.6 + 1986 + 1.2 + 991 + 13.9 + 2 * 105$$

Predicted ET = 10913.7 μs

4.6 MXI

Thus far transferring data from the VME to Sparc station using MXI has not added on any extra overhead event time.

4.7 Predicted Rate Tables

Now that we have a method of predicting event time for a given configuration, one can generate a table of DAQ rates for various configurations.

The following two tables show the maximum event rate the TTR DAQ system can maintain for each configuration. These rates are for MXI, not Ethernet. Table 4.1 shows predicted rates for ADC modules and table 4.2 displays predicted rates for TDC modules. The rates are in Hz, events per second.

	Number of ADC Modules			
	5	10	15	20
Crates				
1	220	202	187	174
2	144	129	117	107
3	107	95	85	77

Table 4.1 : ADC Predictions

	Number of TDC Modules			
	5	10	15	20
Crates				
1	177	162	149	138
2	109	98	88	81
3	79	70	63	57

Table 4.2 : TDC Predictions

A variation on groups was not shown as each crate is read out as a single entity. The above tables reflect only 1 group. If the required number of modules needs to be broken up into separate groups, then an overhead of about 14 μs should be added for each additional group.

If one wants to predict a rate for ethernet, merely multiply the bytes per event by 1.7 μs per byte to get the added overhead. **This multiplier is only good for the Physics Research Snode Ethernet subnet.**

4.8 Concluding Remarks

In summary, the main bottle-necks are the GPIB commands. The READREQ is essentially two GPIB commands a dvwrt() to request the data and a dvrd() to fetch the data. The SETBG, CLRLAM, CLRCRATE, and TCR functions all use dvwrt(). Each of these commands is on the order of 1 ms. Only the WAIT (ibwait()) command at 130 μs is differnet. Because the GPIB command cost such precious time the barebones real-time buffering is almost lost in the noise.

4.8.1 Upgrade Paths for the TTR-DAQ

Given the above conclusions we can:

1) Replace GPIB with CAMAC branch drivers (for instance CAMAC data-highway)
2) Replace CAMAC with VME (VXI)
3) Divide the DAQ system into multiple data paths. A branched system with N DAQ streams would result in approximately 1/ N reduction in read out time.

Currently, design has started on the third alternative mentioned above. Alternatives 1 and 2 are determined by factors external to the DAQ.

Table of Contents
TTR DAQ User's Guide

A. TTR DAQ SYSTEM USER'S GUIDE

This appendix will describe how to use the TTR DAQ system. The real-time system software and the GUI software must both be started. The user just needs some familiarity with using open windows. All figures have been appended to the end of this chapter.

This document is divided into four major sections:

- Initial Load of Software
- Configuring a Run
- Starting and Monitoring a Run
- Spying a Run

A.1 Initial Load of Software

The following steps will take the user from logging into a SPARCstation to starting the TTR DAQ software. The characters in **bold** are the commands the user must type in. All other characters are just output to the screen. It is assummed that the user is going to use the SPARCstation with hostname "stake" as the display station and the SPARCstation with hostname "stone" as the GUI CPU. Other stations can be used for display, but "stone" must be used as the GUI CPU as it is physically linked to the VME crate using the MXIbus interconnect.

stake login: **ttrdaq**
Password: **[Ask a TTR Friend]**
Login messages will appear here...
ttrdaq_stake[51] **openwin3.0**

In a few minutes the user should be in a multiple window environment. Using two of the windows the user can now start the real-time software and the GUI software by typing the following sequence of commands. The first window will be used to login into the real-time OS and load the GPIB device driver and other real-time software. The second window will be used to start the GUI.

WINDOW #1

ttrdaq_stake[51] **rlogin ttrvme**

TTRVME> **<ttrRT**
ld < esp.o
value = 0 = 0x0

ld < ttr.o
value = 0 = 0x0
gpibDrv
value = 0 = 0x0
gpibDevCreate "/gpib0", 0xa000
value = 0 = 0x0
ttr
value = 3900268 = 0x3b836c
TTRVME>
TTR (MXI) Version 1.0.a
Waiting for GUI hook up...

WINDOW #2

ttrdaq_stake[51] **xhost +**
all hosts being allowed
(access control disabled)
ttrdaq_stake[52] **rlogin stone**
Login messages will appear here...
ttrdaq_stone[51] **setenv DISPLAY stake:0.0**
ttrdaq_stone[52] **cd /daq1/ttr/gui**
ttrdaq_stone[53] **ttrStart**

Using printer interrupt...
Connection made with GUI

CONFIGURATION FOR SB-MXI:

Logical Address Configuration:
Logical Address: 0
Device Type: Message Based Device
Address Space: A16 only
Bus Configuration:
System Controller
SB-MXI board and the NI-VXI driver is initialized
Command Connection Made with VME.
Attempt Handshake with VME ...
... Handshake with VME Complete.
Data Connection Made with VME.

If all has gone right, the user should be looking at the TTR control panel (Figure A.1). If (value = -1 = 0xffffffff) appears anywhere in window #1, the MVME147 board may need to be rebooted. To reboot the board either type CNTL-X in window #1 or press the reset button on the board itself. The

next two sections cover different options and default values that can be modified.

A.1.1 ttrStart Options

The ttrStart script starts four processes; ttrServer, dataCollect, tapeCntl and ttrgui. These processes all share a common configuration file "ttr.config". If the user desires to use a different configuration file, say "my.config", one should type "ttrStart -f my.config". The user can also run the GUI in a offline mode (or demonstration mode) by typing "ttrStart -d". This works on any machine as it does not require any communication with the real-time. The user could use this mode to access the Crate Editor without having to start the whole DAQ system. To end a session properly, the user should type ttrStop. This script will cause all processes on both systems to receive shut down messages and execute a graceful shut down.

A.1.2 TTR Configuration File

The user can either modify the ttr.config file or make a copy of this file and use the -f option as explained above. The ttr.config file looks as follows:

```
DATA FILE DESTINATION DIRECTORY        : /daq1/ttr/log/
TAPE DATABASE DESTINATION DIRECTORY: /daq1/ttr/log/
RUN NUMBER FILE NAME                   : /daq1/ttr/run.number
MVM-147 ETHERNET ADDRESS               : 134.3.48.21
DEFAULT FILE SIZE                      : 50048576
DEFAULT MEDIUM                         : 8200
DEFAULT RECORD                         : 2
ZERO SUPPRESSION                       : 0
DATA ACQUISITION MASK                  : 32
TRIGGER MASK                           : 0
STAS MASK                              : 2
NUMBER OF CCDF's                       : 2
CCDF DIRECTORY                         : /daq1/ttr/gui/
CCDF NAME                              : init.ccdf
CCDF DIRECTORY                         : /daq1/ttr/gui/
CCDF NAME                              : test.ccdf
```

The following are accepted values for each of the entries:

```
DATA FILE DESTINATION DIRECTORY        : directory
TAPE DATABASE DESTINATION DIRECTORY: directory
RUN NUMBER FILE NAME                   : directory
MVM-147 ETHERNET ADDRESS               : ip address
```

DEFAULT FILE SIZE	: bytes
DEFAULT MEDIUM	: 8200 or 8500
DEFAULT RECORD	: 0 - write tape and disk, 1 - write tape only, 2 - write disk only, 3 - save no data
ZERO SUPPRESSION	: 0 - off, 1 - on
DATA ACQUISITION MASK	: decimal value of mask
TRIGGER MASK	: decimal value of mask
STAS MASK	: decimal value of mask
NUMBER OF CCDF's	: integer
CCDF DIRECTORY	: directory
CCDF NAME	: file name
... how many directory / name combinations depend on number of CCDF's	
CCDF DIRECTORY	: directory
CCDF NAME	: file name

A.2 Configuring a Run

The control panel (Figure A.1) is the first window the user encounters when the TTR DAQ software is started. To configure and start a run the user must be the master controller of the read out. As will be seen in section A.4 other users can be spying the run. To prevent other users from causing interference in the readout only the master controller has supervisor privileges. To become the master controller, the user must click on the "Master" button on the control panel. The current status message will notifiy the user whether a master controller already exists or that the user has become the master controller. Once the user is the master controller the "Start", "Pause Run", and "Tape" buttons should become active.

Two steps must be taken to configure for a run; first the user needs to enter the Crate Editor and define the read out configuration and second the user needs to enter the Tape Panel and define the data storage medium and initialize it.

A.2.1 Crate Editor (CED)

Selecting the "CED" button on the control panel will yield the main window of the Crate Editor along with the CED Assignment Panel (Figure A.2). From this window the user can load, edit, and save a crate configuration definition file (CCDF) along with assigning configurations to different parts of the read out. The main panel contains two menu buttons; "File" and "Crate" and two other buttons; "Assign Panel" and "View Format". A "Help" button and the file name of the current CCDF is displayed next to the menu buttons. Below the buttons, various messages are displayed to aid the user. The canvas contains a color coded summary of the current configuration, with a color legend at the top. Each crate displayed shows the types of modules by color and is identified by type of crate, transfer mode, transfer bits, ID number, address and whether it is offline or online.

A.2.1.1 CED File Menu

The "File" menu contains six entries; List, Load, Save, Remove, Clear, and Exit. Like any other editor the user must build from scratch or load a file to be edited. The type of file that the CED recognizes is of a special format and is called crate configuration definition file (CCDF). The CED keeps a list of CCDF structures that have been loaded into memory. The current CCDF, is the one that is currently being displayed and edited.

The "List" entry is a pullright menu which reveals a list of CCDF's previously loaded into memory. Selecting a CCDF from the list will cause the current CCDF to be placed in the list and the selected one to become the current CCDF. In a sense the two configurtions just trade places.

The "Load" entry will pop up a load CCDF window. From this window the user should enter the path and file name of the CCDF file that is to be loaded. By clicking on the load button, the current CCDF will be placed in the list as described above and the new file will be read in and become the current CCDF.

The "Save" entry will pop up a save CCDF window similar to the load window. This window is to save the current CCDF to disk. If a path and file name already exist, they will be displayed in the window. However, the user can change them if renaming the file is desired.

The "Remove" entry is a pullright menu which reveals the same list of CCDF's displayed in the "List" entry. Selecting a CCDF here though will free this configuration from memory. Hence the user should take care in saving a CCDF before removing it.

The "Clear" entry moves the current CCDF into the list and clears the current CCDF. This way the user can start from scratch defining a crate configuration.

The "Exit" entry not only closes the CED main window but cleans up any other windows that were left open by the user. Also, the current CCDF is moved to the list.

A.2.1.2 CED Crate Menu

The "Crate" menu contains four entries; Goto, Add, Remove, and Probe. The user can easily add or remove crates from the configuration or display a crate window to edit. The user may also query the hardware itself to see what modules are out there using the probe.

NOTE: The probe option works only for TDC and ADC modules, other module types are not recognized due to CAMAC limitations.

The "Goto" entry is a pullright menu which allows the user to select a crate to edit using a crate window (Figure A.3). From this window the user can edit parameters at the crate level and the module level. At the crate level the user can define the transfer mode (Single Transfer, Address

Scan, Q Stop Scan, Q Repeat Scan), the transfer bits (24, 16, 8), the address and turn the the crate online or offline.

NOTE: If a crate is turned offline, then it will not be considered in a probe or in the event format structure. So even though the crate may still physically be daisy chained to the other crates, an offline crate will not be read out. Hence this is a software switch.

At the module level the user may edit the module type and the technology type for any given station in the crate by making the appropriate selections in the "Module" menu and "Technology" menu and then clicking inside the boundary of that particular station. It is important to remember that both the module type and technology type are assigned at then same time so both must be properly selected before clicking on any station. To remove a module from a station (slot), the user should choose "Empty" from the Module menu and "None" from the Technology menu and then click on that slot.

Finally, the user can define upper and lower threshold values for a module by clicking the "Threshold" button. The window operates in two different modes; definition and threshold mode. The user can toggle between these two modes by clicking on the same button. The definition mode was just described and now the threshold mode (Figure A.4) will be explained.

The user may modify the threshold values for the whole crate, a single module or a single channel of a module. To modify a single channel the user needs to click the right mouse on that channel. A threshold window will appear allowing the user to make the changes. To modify all the channels of a module, the user needs to click on the empty box below the last channel. To modify the entire crate, the user needs to click on the crate controller.

The "Probe" entry results in the GUI system commanding the real-time system to query the crates for the actual configuration. This works quite well and is convenient when the user is not in close proximity to the crates. The only draw back found so far is that the trigger module described in section 2.1.2 reports back as an empty station. So the user would have to manually configure this station.

A.2.1.3 CED Assignment Panel

The TTR DAQ System has defined the following types of events:

- Normat Event : Externally Triggered Read out of Crates
- Special Event: Inital and Last Read out of Status Crate
- Special Comment: Run Information
- Threshold Events: Run Information
- Calibration Comment: Run Information
- Voltage Event: Run Information

The "Assign Panel" button brings up the CED Assignment Panel. This panel is brought up by default when the CED is started. For each of the event types mentioned above the user can select "yes" or

"no" whether the user wants these events as part of the run or not. Care needs to be taken when selecting the normal event and the special event, as the panel also assigns the crate configuration for those events. The panel uses the current CCDF loaded, therefore the user needs to load another CCDF in between assigning the normal and special events.

Also on this panel is a "Trigger Mask" menu for selecting the proper trigger and a "STAS Plane Configuration". This value is used as a bit pattern to determine which chains in the STAS module need to be initialized. For example a value of 37 (or 0x25) will initialize chains 0, 2, and 5.

A.2.1.4 CED View Format

The "View Format" button opens a window (Figure A.5) which displays the current CCDF event format structure along with a predicted rate (See Chapter 4).

A.2.1.5 CED Help

This menu is for future use.

A.2.2 Tape Configuration

Selecting the "Tape" button will yield the tape configuration subwindow (Figure A.6). The user can then select where the data should go using the data record options; write tape and disk, write tape only, write disk only, or save no data. If the user decides to save data to tape, then a tape should be loaded in the tape drive. Next the user needs to select the save format whether it be exabyte 8200 or 8500. If this tape had been used in a past run, then a tape header file should already exist and the user should just have to click on the "Online" button. Otherwise, if this tape is being used for the first time it should be formated first by clicking on the "Format Tape" button. This will automatically put the tape online. Once online, a data base file name should appear in the panel along with a tape device number. The user should set the zero suppression flag on if this option is desired.The user may also select the size of the "raw" data files to be stored by using the slider. During a run the user can observe the "Tape Usage" gauge on the control panel to see when a new tape is needed. **The run should be paused or stopped while changing tapes.**

A.3 Starting and Monitoring a Run

Once the crates and tape are configured the user is ready to start a run. By clicking the Start button, the user sends the event format structure information to the real-time system. Once the real-time system initializes the data structure and crates it will send a response back. If all went well, a "Run Started" message will appear in the control panel. This button will then have a new label which will be "Stop". Selecting Stop will send the stop command to the real-time causing the read out to terminate. Once the data buffers have been flushed out the "Run Stopped" message will appear and the button label will revert back to Start.

NOTE: If the Special Comment Event was selected to be sent, then the special comment window will pop up when the "Start" button is clicked. The user must then make any modifications needed and save the special comment file before the start sequence will complete.

Also, two script files are executed when starting and stopping. The script ".ttrBegin" and ".ttrEnd" will be executed when starting and stopping respectively. This provides a manner in which other programs can be initiated and syncronized with the run. For example "ttr2d", the 2D event display, can be started along with other online analysis tools. It is suggested that the user put clean up scripts in ".ttrEnd" to properly clean up any tools started in ".ttrBegin".

Another feature of the TTR DAQ system that should be mentioned here is that the system will automatically stop and start new runs once the raw data file size reaches the user selected limit in the tape configuration panel. A warning message will be issued well before this happens. An automatic start will not re-execute the ".ttrBegin" script and will not bring up the special comment window. This prevents the system from being stopped during an all night run.

The "Pause" button, differs from stopping in that the read out is just paused indefinitely from looking for the next trigger. The data buffers will continue to flush out though. After the system is paused the button will get a new label, "Continue". Selecting this will cause the read out routine to be released and to start processing events again.

Right below the main buttons on the control panel the user will find the current system message. Below this mesage one can observe the current event number, run number, read out frequency, tape usage meter (in percent), accumulative dead time and instantaneous dead time. On the bottom of the control panel are 3 columns of error and status mesages. The first column displays error messages strictly from the real-time system. The second column displays errors occuring in the rest of the DAQ system. The third column is a scrolling column of status messages. When the control panel receives another mesage it is placed in the current status slot, while the rest of the messages are pushed down the stack with the oldest message being thrown away. Each column has a "Clear" button which when clicked will clear all messages in that column.

Once the run is started, three buttons become active on the control panel; Comment, Dump and Show. The next three sections describe these subwindows.

A.3.1 Event Comment Window

Once the run has been started the "Comment" button on the control panel becomes active and the user may enter the event comment window (Figure A.7), where the user can type in a text comment and send it to the real-time portion of the read out to be inserted in the event data as a comment event.

A.3.2 Event Dump Window

Once a run has been started the "Dump" button on the control panel becomes active and the user may enter the event dump subwindow (Figure A.8). This window has three buttons on it's panel. The first button, Dump Event, makes a request to the real-time system to pass back the next event. Once the event is received it is displayed in hexadecimal format with a space between every 16-bit word. By clicking the right mouse button on any word, the contents will be displayed in the panel along with a description of the contents. For example, the message might say "Group Data(3): Module=5, Channel=9, Data=598". This would mean the user clicked on a data cell in group 3.

Since this is a text window if the user types at the keyboard characters will be added to the event dump. If this happens accidently, the user can use the button "Restore Event" to redisplay the current event. If the user would like to display the event that was captured previously, use the "Previous Event" button.

A.3.3 Event Show Window

Once a run has been started the "Show" button on the control panel becomes active and the user may enter the event show subwindow (Figure A.9). This window allows a user to examine the data from a particular slot (module) as it displays a trace of all channels in the module. The user can select the group number and slot (module) number of the station desired.

A.4 Spying a Run

If a user just desires to check the status of the current run, the user can follow the same steps for window #2 in section A.1. This will initiate another GUI with limited privileges but at least the user can request event dumps and check the status of the current read out as messages, tape usage, etc will be updated on all GUI's that are active. The user simple needs to type ttrStart to initiate another GUI.

WARNING: Do not use the ttrStop script to quit out of the window as this will shut down the entire DAQ system. Use the "Quit" entry in the X-window menu by pressing on the right mouse button while the cursor is on the title bar. It is suggested that the user use their own account when spying rather than using the ttrdaq account.

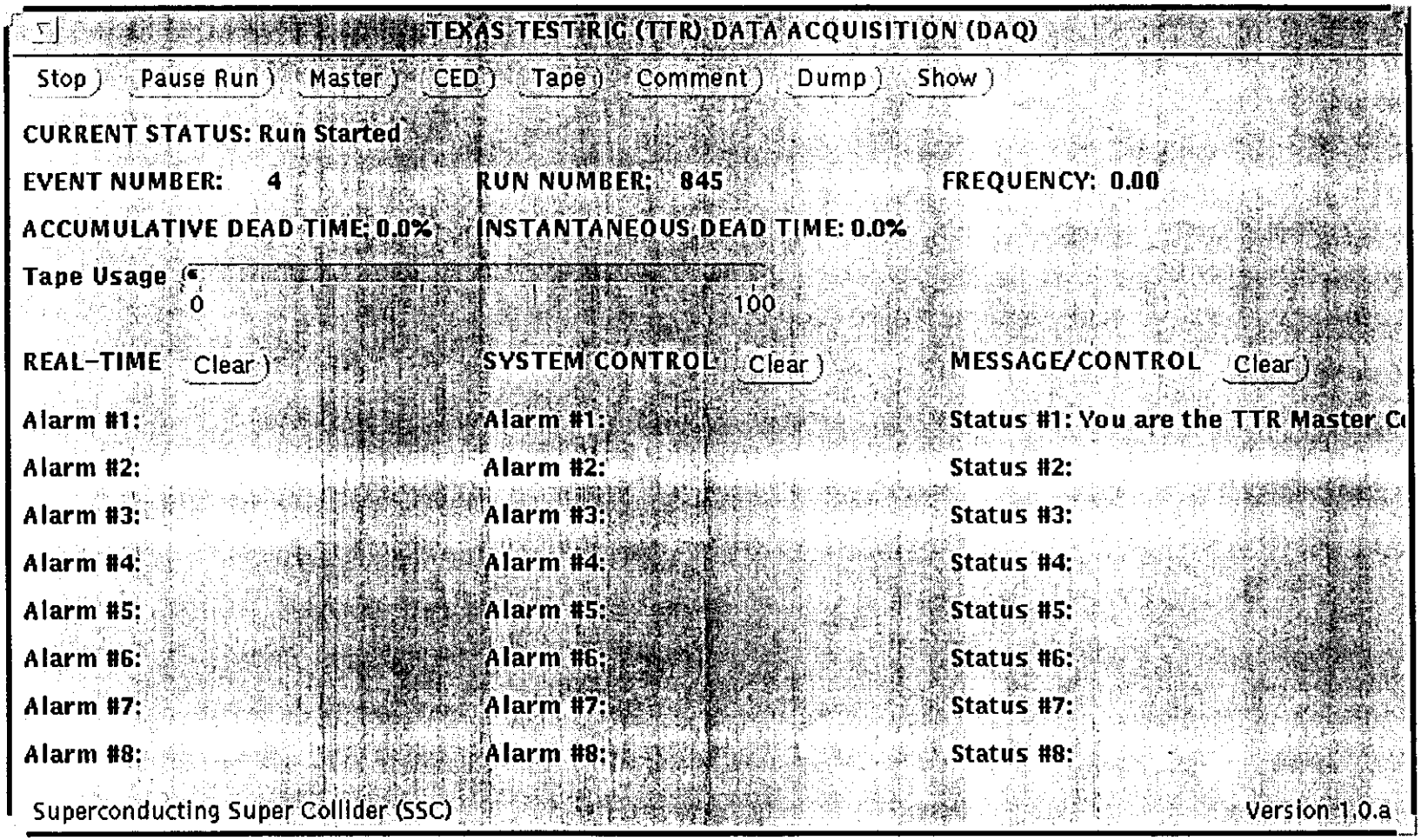


Figure A.1 TTR Control Panel

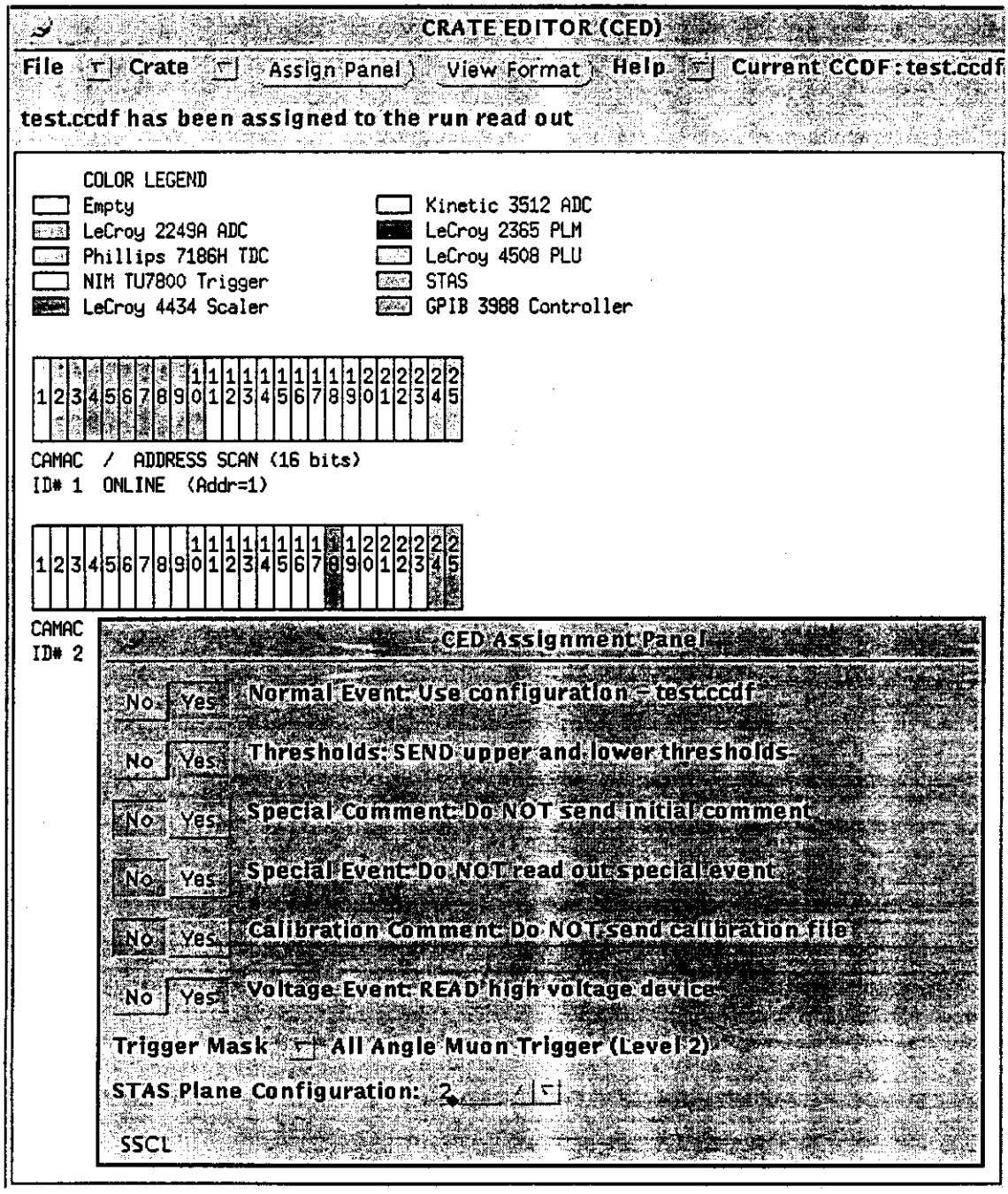


Figure A.2 Crate Editor

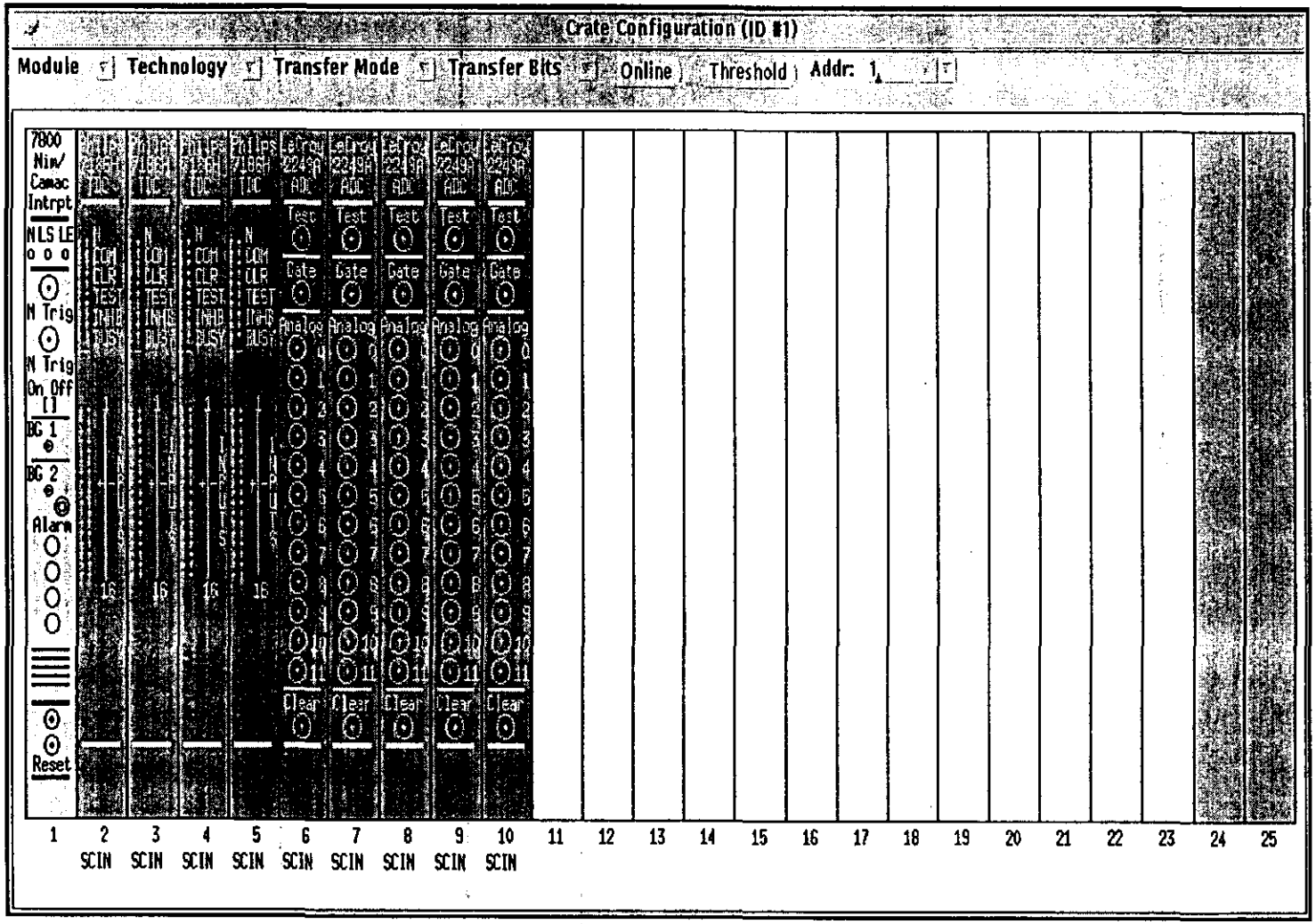


Figure A.3 Crate Definition

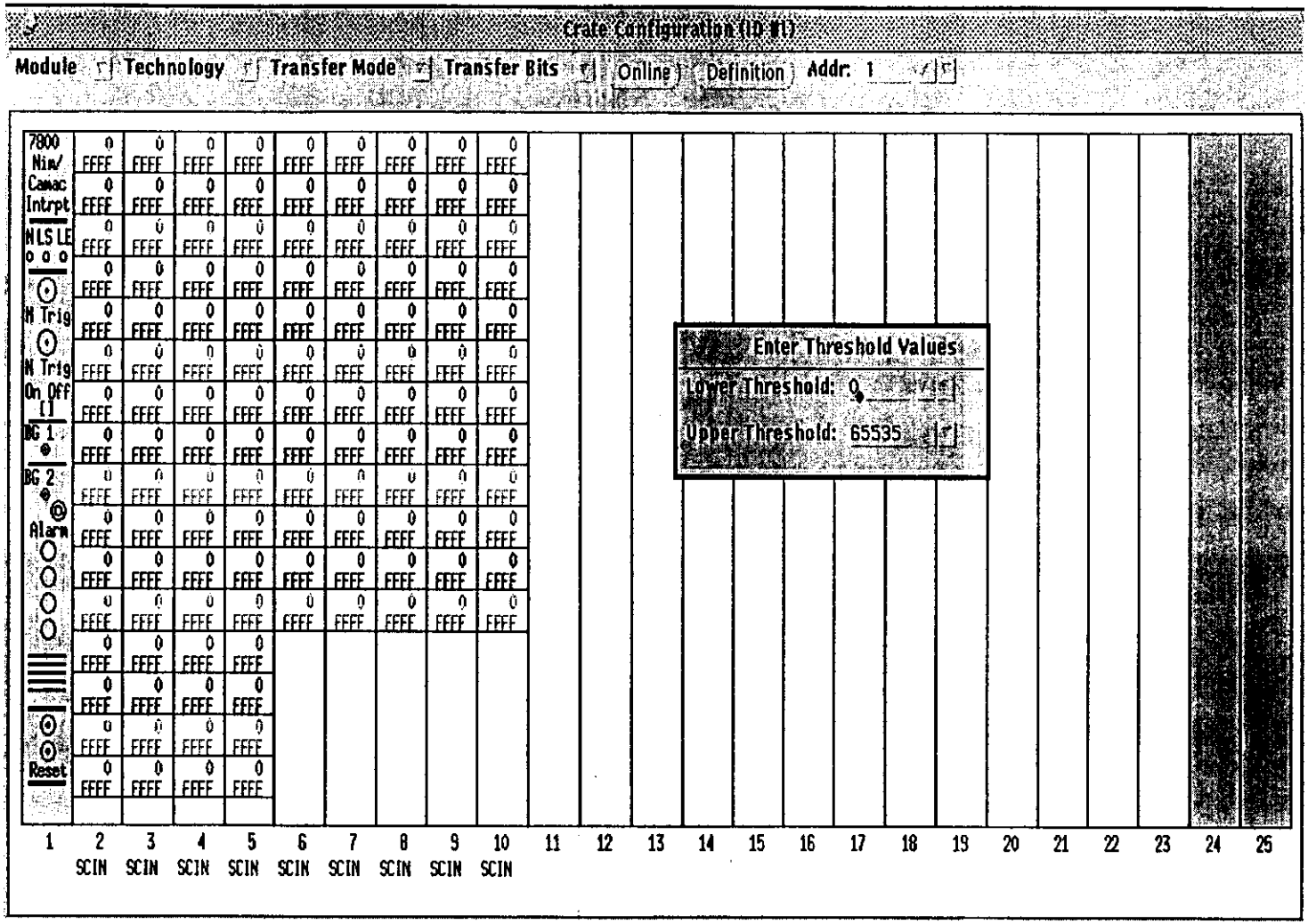


Figure A.4 Threshold Mode

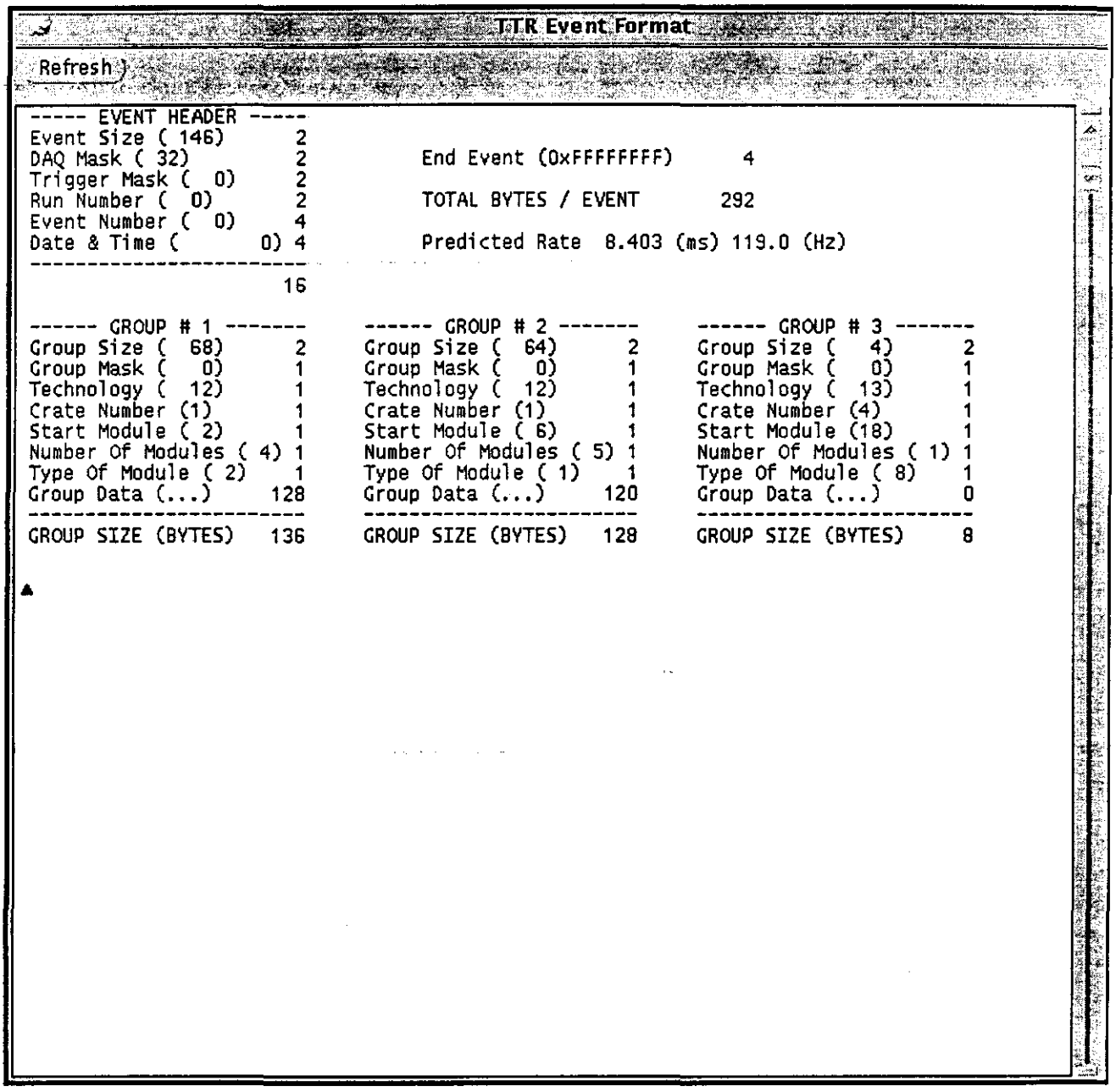


Figure A.5 Event Format

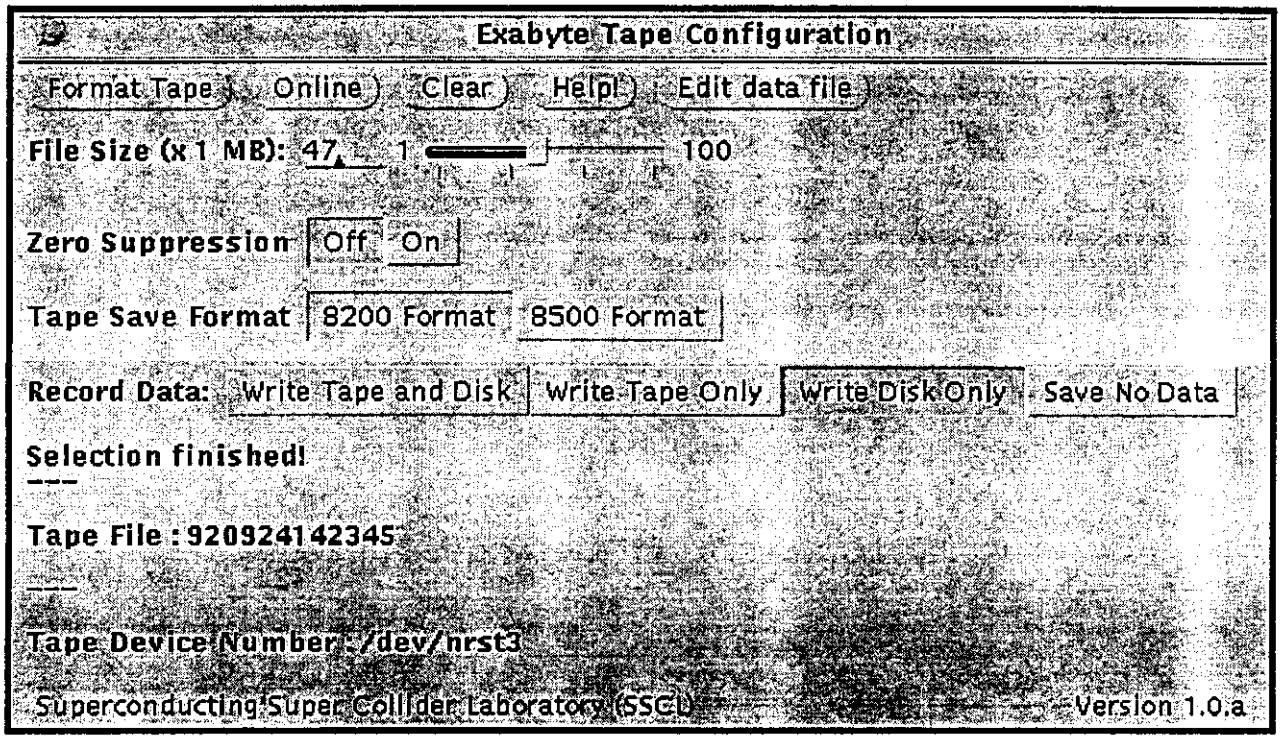


Figure A.6 Tape Configuration

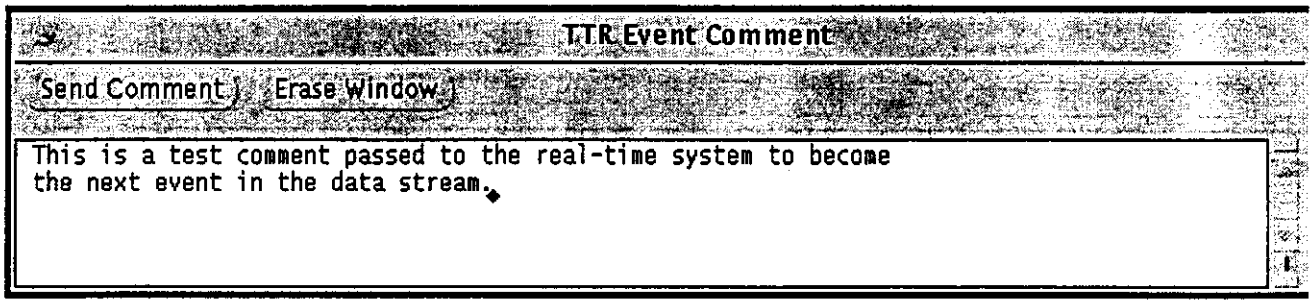


Figure A.7 Event Comment

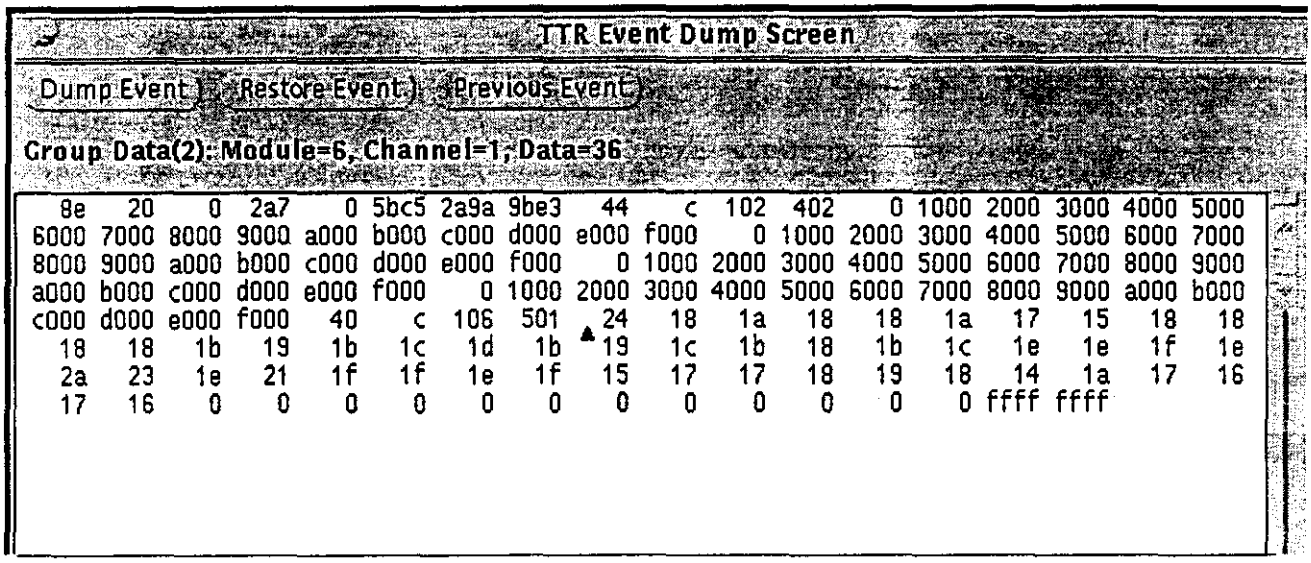


Figure A.8 Event Dump

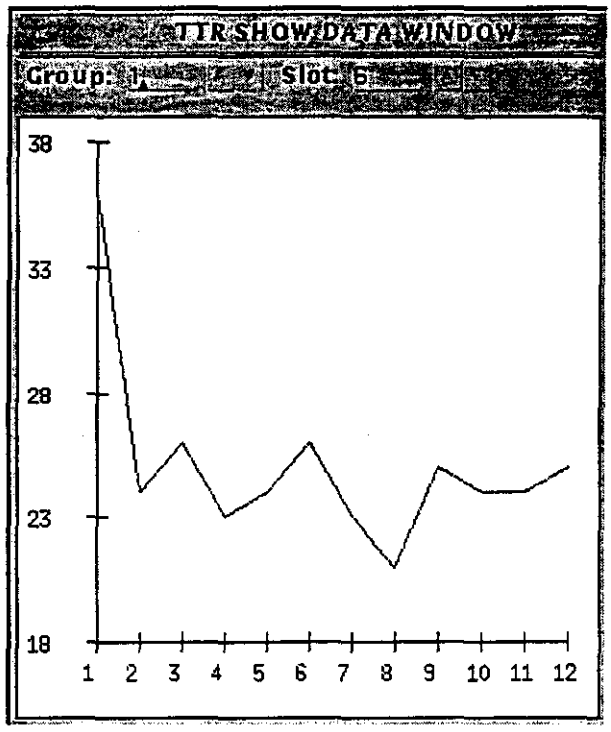


Figure A.9 Show Data