CLAS-NOTE 92-001

CLAS Raw Event Format

L. C. Dennis

Department of Physics

Florida State University

Tallahassee, Florida, 32306

January 8, 1992

Abstract

This document provides specifics of the CLAS raw event format. This format is used for events entering the data analysis stream from tape or from the online event stream after removing zero's and reordering the data. As the first version of the raw event format, it is expected to change with later versions. The form of event files are also described. The structure of the event data, the run header, trailers and checkpoints all adhere to CEBAF common event format but do not make extensive use of the structured format possible within that scheme. The output of these events to physical devices is also described.

I. Introduction

This document outlines all of the details needed to produce a "run" file in the standard CEBAF and CLAS format. In includes all the items which make up a run and all of the details which are needed for detector and component numbering schemes. The items are described in the order in which they occur on the run file. That order normally consists of a run header, followed by events, followed by run checkpoints and finally followed by a run trailer. For details on the definition method see CLASNOTE-91-010[1]. Briefly, the structure of the run is described using ='s to equate data entities, +'s to concatenate data entities, { }'s to denote iterated data entities and ( )'s to denote optional data entities. In this way complex data entities may be broken down into their component parts, and the parts broken down further until only simple entities remain. Data entities used in any one definition which are simple entities are described in the text following the definition. In this way the entities within the run are combined and then defined to provide a complete description of the run file. Tables summarizing these definitions are given at the end of the text. These tables show the order of data words within a run file and the structure of the banks in which they are imbedded.

Also described in this document is a program which interprets runs, allows the user to perform a detailed examination of the event data in them and can be used to convert event data into an ASCII format.

The run format used in this version conforms to the CEBAF common event format. In all cases the desired event data is packed into a single "bank" which contains the CEBAF standard event header, followed by the data. Since these events are to be used at the start of the event analysis stream, they events assume that zero suppression of ADC's and TDC's has already occured and that the event data is grouped by sector, package, subpackage and component. This version of the run files is intended to be the first version. One major difference between this version and the expected final version, is that this version does not take advantage of the expected range of the data values to reduce the event size and contains some redundant information. The size and structure of the events will be optimized in later versions. The banks for the run header, run checkpoints, run trailer and events are then packed into logical records, with no gaps between. The bank for any one of these entities may cross a logical record boundary at any point in the bank.

Unless otherwise stated, the values contained in the file are INTEGER*4 variables. Any real numbers are REAL*4. For character strings there are some standard size conventions. All filenames are 80 characters long. All names are 32 characters long and all other character string lengths are given with their definitions, but in all cases are multiples of 4 bytes. Because of these conventions the event files are relatively long.

In the *version 1* scheme all variables are converted to integers before output. These **integer buffers** are then output using **unfomatted writes**. The **standard size** for the buffer is **32768** integer*4 words. The method by which these conversions are

done is included at the end of the run file summaries. This method has only been tested on VAXstations and DECstations.

II. Event File Definition and Structure

The CLAS event files are expected to hold events and basic information about the run. As such they contain four distinct entities. These are the events, the run header, the run checkpoints and the run trailers. The latter three entities contain similar types of information about the situation under which the event data was produced and scaler information. Each of these four types of data blocks is imbedded in a unique data bank. The generalized bank format is described in the CEBAF Online Data Acquisition Manual[2].

The following provides the shorthand definition of an event file. Each of the entities within the definition is a complex data object which will be described in detail later. Note that the event file definition allows for a run without any run trailer or checkpoints. In general such a run would not be usable, but it has been included here for completeness.

Event File = Run Header
+ { Event }
+ ({ Run Checkpoint
+ ({ Event }) }
+ Run Trailer)

II.A. Run Header

The run header provides all the information needed to begin the processing of a run, plus some general information which is useful for tracking down problems which occurred during a run. The first part of the Run Header, Run Checkpoint and the Run Trailer are the same. The main purpose of the checkpoints and trailers are to serve as "scaler events". They provide the information needed to reconstruct what happened during a run.

Run Header = Header Flag
+ Header Length
+ Event Count
+ Date
+ Time
+ Run Version Number
+ Processing Status Flag
+ Run Description Block
+ Event File Locator List
+ Run Setup Block
+ Scaler/Parameter Setup Block

Header Flag = A single word used to identify this as a Run Header. The standard header flag value is −1111. This information is redundant and will be dropped from future versions.

Header Length = Length (in 4 byte words) of this event header block. This value counts all words in the header.

Event Count = The number of events in the run prior to this point. If this run does not span multiple tapes, the event count in the run header should be zero.

Date (CHARACTER*8) = Current Date in DDMMMYY format followed by 1 fill blank.

Time (CHARACTER*8) = Current Time in HHMMSS format using standard miltary time followed by 2 fill blanks.

Run Version = A software version flag for this run. The current version (described in this document is 1).

Processing Status Flag = A software flag which indicates the state of processing for this run. Since this document describes only unpacked events this flag should always have the value 1.

Scaler/Parameter Setup Block

```
Scaler/Parameter Setup Block  =  Number of Groups
                              +  { Group Name
                              +    Number of Scalers
                              +    Number of Parameters
                              +    { Scaler Name }
                              +    { Parameter Name } }
```

Number of Groups = Number of scaler/parameter groups defined by this block. When this value is zero, no groups follow.

Group Name (CHARACTER*32) = A character string used to assign a name to this scaler/parameter group.

Number of Scalers = Number of scalers defined by for this group. When this value is zero, no scaler names follow.

Number of Parameters = Number of parameters defined for this group. When this value is zero, no parameter names follow.

Scaler Name (CHARACTER*32) = A character string used to name this scaler.

Parameter Name (CHARACTER*32) = A character string used to name this parameter.

Run Description Block

Run Description Block = Experiment Name
+ Run ID Number
+ Beam Type
+ Beam Energy
+ Target Type
+ Run Purpose
+ Number on Shift
+ { Person on Shift }

Experiment Name (CHARACTER*32) = Identifying name of an experiment.

Run ID Number = Number which defines this run in the sequence of runs for this experiment.

Beam Type = Type of beam used for this run (electron, photon, polarized, etc.) (0 - photon beam, 1-electron beam and 2 - polarized electron beam).

Beam Energy (REAL*4) = Energy (in GeV) of beam used for this run.

Target Type = Type of Target used in this run. Integer value = 1000*Z + A of the targe nucleus.

Run Purpose (CHARACTER*256) = Description of the goal of this run.

Number on Shift = Number of experimenters on shift for this run.

Person on Shift (CHARACTER*32) = Name of an experimenter on shift.

Run Setup Block

This block points to all the data needed to analyze the run, including the details of the trigger and data acquisition software used. It also points to the CLAS Configuration data based needed to reconstruct the event.

Run Setup Block	=	Trigger Specification Locator
	+	Data Acq. Specification Locator
	+	CLAS Configuration Locator
	+	User Configuration Locator

Trigger Specification Locator (CHARACTER*80) = Generalized "filename" where information on the setup of the trigger is stored. For the present we will assuming we will adopt a file naming convention which makes it easier to uniquely identify a file. The special name "ONTAPE" will be used to denote that the run is on a tape. Similar names will be given to other offline storage media. The string contains a description of the file in a compressed format as given below.

For a file on a computer:

machine name :: *diskname* :[*directory*] *filename*

This format will be used even if the file is on a unix machine. In all cases this locator name is translated to the name actually needed.

For a file on a tape:

ONTAPE :: *location* :[*medium type* . *tape name*] *filename*

(Ex. TSL = FSULCD::USER:[LARRY.GAMMAP]EXP128.TSL or if it is on tape then TSL = ONTAPE::CEBAF:[6250.TAPE001]EXP128.TSL).

Data Acquisition Specification Locator = Generalized filename where information on the setup of the data acquisition system is stored. (See note above for the format).

CLAS Configuration Locator = Generalized filename where information on the setup of the CLAS configuration database (geometry, status and calibration information) is kept. (See note above for the format).

User Configuration Locator = Generalized filename where information on the user defined setup is kept. (See note above for the format).

Event File Locator List

Event File Locator List	=	Event File Locator Count
	+	{ Event File Locator }

Event File Locator Count = Is the number of event file locators contained in this list

Event File Locator = The generalized filename of a file containing event data for this run. The generalized file name includes information on the computer and type of device used. (See note on preceeding page).

Event Format

In general events can be input or output at different stages of the event analysis process. The status of the analysis is indicated by the **Processing Status Flag** in the Run Header. The event format described here is only for unpacked events. However, this format is compatable with all of the other event types. It is expected that events will be extracted from the event buffer before being used and that the event buffer itself will not be equivalenced.

The structure of events follows that of the detector from event level data, to sector level data, to detector package level data, to detector subpackage level and finally to detector component level data. For some detector packages or some applications it is more convenient to ignore the subpackage representation of the data. This fact will not be represented in the event format but will be dealt with automatically in the event database routines.

Event = Event Header Flag
+ Event Length
+ Event Type
+ Event Sector Count
+ Event ID
+ { Sector }
+ Event Description Data

Event Header Flag = An integer flag which indicates that this is an event. The standard value for this flag is -4444.

Event Length = The length of this event in words.

Event Type = This is meant to be an integer value coming from the trigger system. Thus it would normally be between 1 and 8. Since the meaning of these values will vary from run to run there is no standard value for this number.

Event Sector Count = The number of sectors in this event.

Event ID = This number (actually implemented as two numbers) provides an ID number for the event. This number can be used as a reference to the event.

Event Description Data = This block is not used by unpacked data, so it should be empty. It would normally be used for data which comes from information from more than one sector.

Sector

This section of the event data contains all the information pertaining to one sector of the detector.

Sector = Sector ID
+ Sector Length
+ Sector Package Count
+ { Package }
+ Sector Description Data

Sector ID = An integer flag which indicates the ID number of this sector. These numbers run from 1 to 6 for the CLAS sectors, the photon tagger has been assigned sector id number 7.

Sector Length = The length of this sector in words.

Sector Package Count = The number of packages in this sector.

Sector Description Data = This block is not used by unpacked data, so it should be empty. It would normally be used for data which comes from information from more than one package within this sector.

Package

This section of the event data contains all the information pertaining to one detector package in one sector of the detector.

Package	=	Package ID
	+	Package Length
	+	Subpackage Count
	+	Unpacked Component Size
	+	Calibrated Component Size
	+	{ Subpackage }
	+	Package Description Data

Package ID = An integer flag which indicates the ID number of this package. The same package ID's are used in all sectors. These numbers must be combined with the sector ID to produce a unique value.

1	Reg. 1 Axial Superlayer	2	Reg. 1 Stereo Superlayer
3	Reg. 2 Axial Superlayer	4	Reg. 2 Stereo Superlayer
5	Reg. 3 Axial Superlayer	6	Reg. 3 Stereo Superlayer
7	Scintillation Counters	8	Cerenkov Counters
9	Front of Forward Calorimeter	10	Rear of Forward Calorimeter
11	Front of Backward Calorimeter	12	Rear of Backward Calorimeter
13	Trigger System	14	Tagger Energy Scintillators
15	Tagger Timing Scintillators	16 ···	Others as needed

Package Length = The length of this package in words.

Subpackage Count = The number of subpackages in this detector packages. There are several dectector systems which contain subpackages. Each drift chamber superlayer (1 superlayer = detector package) contains 6 subpackages (1 subpackage = 1 layer). Each Electron Calorimeter (1 package = 1 calorimeter) contains 6 subpackages, (1 subpackage = 1 plane). They are u-front plane, v-front plane, w-front plane, u-rear plane, v-rear plane and w-rear plane. Each cerenkov package contains two subpackages (high-ϕ detectors and low-ϕ detectors). The number schemes for these subpackages are from inside to outside and then from low-ϕ to high-ϕ.

Unpacked Component Size = The number of parameters included in the unpacked for this component.

Calibrated Component Size = The number of parameters included in the unpacked for this component.

Package Description Data = This block can be used by unpacked data. So it should contain a parameter count followed by the given number of integer data values. In particular the Electron Calorimeter produces a shower sum signal which will be the stored here. The order of this data is: Number of Data Values (=3), Front Shower Sum, Rear Shower Sum and Combined Shower Sum.)

Subpackage

This section of the event data contains all the information pertaining to one of the subpackages.

Subpackage	=	Subpackage ID
	+	Subpackage Length
	+	Subpackage Component Count
	+	{ Component }

Subpackage ID = An integer flag which indicates the ID number of this subpackage. These number are unique within a detector type but are repeated for different sectors. (for example: Drift chamber layers have id's running from 1 to 36).

Subpackage Length = The length of this subpackage in words.

Subpackage Component Package Count = The number of detector components in this subpackage. For drift chambers components are single wires. For the scintillators each scintillator is a component. For the cerenkov counters each phototube is a component. For the electron calorimeter each phototube is a component.

Component

Components consist only of a component ID and any parameters read from the electronics for these components. The order for the parameters is show below.

Component Type	Param. Count	Parameter Order
Drift Chamber Wire	3	ID, ADC, TDC
Scintillator	5	ID, low-ϕ ADC, low-ϕ TDC, high-ϕ ADC, high-ϕ TDC
Cerenkov Phototube	3	ID, ADC, TDC
EC Phototube	3	ID, ADC, TDC
Tagger Timing PT	3	ID, ADC, TDC
Tagger Energy PT	3	ID, ADC, TDC

The numbering scheme for all components starts at 1 for the innermost, most forward angle. The component ID increases as one moves outward and is reset for each subpackage. The table below summaries the maximum number of components in each subpackage at this time.

Name	Count
DC layers 1 - 12	128
DC layers 13 - 36	192
Scintillators	48
Cerenkov low-ϕ	36
Cerenkow high-ϕ	36
U-Front	66
V-Front	66
W-Front	66
U-Rear	66
V-Rear	66
W-Rear	66

Run Header Format Summary

This summary contains references to large blocks of data which are described immediately following the Run Header Format Summary Table.

Pointer	Description
Run Header Start $\longrightarrow$	Run Header Flag ($= -1111$)
	Header Length
	Event Count ($= 0$)
	Date_1
	Date_2
	Time_1
	Time_2
	Run Version Number ($= 1$)
	Processing Status Flag ($= 1$)
	$\text{User Expansion Space}_1$
	$\cdots$
	$\text{User Expansion Space}_8$
RDB Start $\longrightarrow$	Experiment Name_1
	$\cdots$ (Run Description Block)
RDB End $\longrightarrow$	$\text{Experimenter Name}_{N_B,8}$
EFLL Start $\longrightarrow$	Event File Count $= N_{EF}$
	$\cdots$ (see Event File Locator List)
EFLL End $\longrightarrow$	$\text{Event File Name}_{N_{EF},8}$
RSB Start $\longrightarrow$	$\text{Trigger Spec. Locator}_1$
	$\cdots$ (see Run Setup Block)
RSB End $\longrightarrow$	$\text{CLAS Config. Locator}_8$
SPSB Start $\longrightarrow$	Number of Groups $= N_g$
	$\cdots$ (see Scaler/Param. Setup Block)
SPSB End $\longrightarrow$	$\text{Parameter Name}_{N_g,N_p,8}$

Run Description Block

Pointers	Description
RDB Start $\longrightarrow$	RDB Block Length
	Experiment Name$_1$
	...
	Experiment Name$_8$
	Run ID
	Beam Type
	Beam Energy
	Target Type
	Run Purpose$_1$
	...
	Run Purpose$_{64}$
	Number on Shift = N_E
	Experimenter$_{1,1}$
	...
	Experimenter$_{N_E,8}$
	RDB User Expansion Space$_1$
	...
RDB End $\longrightarrow$	RDB User Expansion Space$_{16}$

Event File Locator List

Pointers	Description
EFLL Start $\longrightarrow$	Event File Count = N_{EF}
	Event File$_{1,1}$
	...
EFLL End $\longrightarrow$	Event File$_{N_{EF},8}$

Run Setup Block

Pointers	Description
RSB Start $\longrightarrow$	Trigger Spec. Locator_1
	...
	Trigger Spec. Locator_{20}
	Data Acq. Spec. Locator_1
	...
	Data Acq. Spec. Locator_{20}
	CLAS Config. . Locator_1
	...
	CLAS Config. Locator_{20}
	User Config. Locator_1
	...
RSB End $\longrightarrow$	User Config. Locator_{20}

Scaler, Parameter Setup Block

Pointers	Description
SPSB Start $\longrightarrow$	Number of Groups = N_g
	Group Name $_{1,1}$
	...
	Group Name $_{1,8}$
	Number of Scalers$_1$ = N_s
	Number of Parameters$_1$ = N_p
	Scaler Name$_{1,1,1}$
	...
	Scaler Name$_{1,1,8}$
	Scaler Name $_{1,2,1}$
	...
	Scaler Name $_{1,N_s,8}$
	Parameter Name$_{1,1,1}$
	...
	Parameter Name$_{1,1,8}$
	Parameter Name $_{1,2,1}$
	...
	Parameter Name $_{1,N_p,8}$
	Group Name $_{2,1}$
	...
	Group Name $_{N_g,8}$
	Number of Scalers$_{N_g}$ = N_s
	Number of Parameters$_{N_g}$ = N_p
	...
SPSB End $\longrightarrow$	Parameter Name $_{N_g,N_p,8}$

Raw (or Unpacked) Event Format Summary

Pointer		Description
Event Start	$\longrightarrow$	Event Header ($= -4444$)
		Event Length
		Event Type
		Event Sector Count ($= N_s$)
		Event ID_1
		Event ID_2
		Event Expansion $Space_1$
		...
		Event Expansion $Space_5$
Sector $Start_1$	$\longrightarrow$	Sector ID
		Sector Length ($= L_{s_1}$)
		Sector Package Count ($= N_p$)
Package $Start_1$	$\longrightarrow$	Package ID
		Package Length ($= L_p$)
		Subpackage Count ($= N_{sp}$)
		Unpacked Component Size ($= UNP_P$)
		Calibrated Component Size ($= CAL_P = 0$)
Subpackage $Start_1$	$\longrightarrow$	Subpackage ID
		Subpackage Length ($= L_{sp_1}$)
		Component Count ($= N_c$)
Component $Start_1$	$\longrightarrow$	Component ID = ($Parameter_{1,1}$)
		$Parameter_{1,2}$
		...
Component End_1	$\longrightarrow$	$Parameter_{1,UNP_p}$
Component $Start_2$	$\longrightarrow$	Component ID = ($Parameter_{2,1}$)
		...
Subpackage End_1	$\longrightarrow$	$Parameter_{N_c,UNP_p}$
Subpackage $Start_2$	$\longrightarrow$	Subpackage ID
		...
Subpackage $End_{N_{sp}}$	$\longrightarrow$	$Parameter_{N_c,UNP_p}$
		Package Description Parameter Count $= N_{PDP}$
		Package Description $Parameter_1$
		...
Package End_1	$\longrightarrow$	Package Description $Parameter_{N_{PDP}}$
Package $Start_2$	$\longrightarrow$	Package ID
		...
Package End_{N_p}	$\longrightarrow$	Package Description $Parameter_{N_{PDP}}$
Sector $Start_2$	$\longrightarrow$	Sector ID
		...
Event End	$\longrightarrow$	Package Description $Parameter_{N_{PDP}}$

Events from the front end electronics, following reorganization by sector, package, component, etc. have the structure given above. These events, if no other analysis is done, are the events which get written to tape.

Event Bank Structure

The raw events are embedded in a single CEBAF event structure, so that the order of the words in the event is a follows.

Pointer	Word	Description
Bank Start $\longrightarrow$	1	Bank Length - 1
	2	Bank Tag (= 'CEBA1001'X)
Event Header Start $\longrightarrow$	3	Event Header Length - 1 (= 4)
	4	Event Header Tag (= 'C0010100'X)
	5	Event ID
	6	Event Classification (= '000CEBAF'X)
Event Header End $\longrightarrow$	7	Event Status (0 for now)
Event Data Start $\longrightarrow$	8	Event Data Length - 1
	9	Event Data Tag (= '800B0100'X)
	10	Event Data as described above
	11 $\cdots$	$\cdots$
Event Data End $\longrightarrow$	Bank Length	Event Data

II.C. Run Checkpoint

The run checkpoint is an optional complex data entity which serves as an intermediate output of run statistics, and scaler and parameter output. See the Run Header for more details on some of these values.

Run Checkpoint = Checkpoint Flag
+ Checkpoint Length
+ Event Count
+ Date
+ Time
+ Scaler/Parameter Value Block

Checkpoint Flag = A single word used to identify this as a Run Header. The standard Checkpoint Flag = -2222.

Checkpoint Length = Length (in words) of this Checkpoint block.

Event Count = The number of events in the event prior to this point.

Date = Current Date (see Run Header).

Time = Current Time (see Run Header).

Run Checkpoint Data Summary

Pointer	Description
Checkpoint Start $\longrightarrow$	Checkpoint Flag (= −2222)
	Checkpoint Length
	Event Count (= 0)
	Date_1
	Date_2
	Time_1
	Time_2
SPVB Start $\longrightarrow$	Number of Groups = N_g
	$\cdots$ (see Scaler/Param. Value Block)
SPVB End $\longrightarrow$	Param. Value_{N_g,N_p}

Run Checkpoint Bank Structure

The run checkpoints are contained in a single bank structure. The additional CEBAF event format data is defined below. Note the abbreviation of checkpoint as CHKPT.

Pointer	Word	Description
Bank Start $\longrightarrow$	1	Bank Length - 1
	2	Bank Tag (= 'CEBC1001'X)
CHKPT Data Start $\longrightarrow$	3	CHKPT Length - 1 (= Bank Length - 3)
	4	CHKPT Tag (= '800B01000'X)
	5	CHKPT Data
	6 $\cdots$	$\cdots$
CHKPT Data End $\longrightarrow$	Bank Length	CHKPT Data End

Scaler/Parameter Value Block

Scaler/Parameter Value Block = Number of Groups
+ { Number of Scalers
+ Number of Parameters
+ { Scaler Value }
+ { Parameter Value } }

Number of Groups = Number of groups defined in this block. This must match the number and order of the groups in the Scaler/Parameter Setup Block.

Number of Scalers = Number of scalers defined for this group. If this number is zero, no scaler values follow. However, if this number is non-zero, it must match the number for the corresponding group in the Scaler/Parameter Setup Block and the scaler values must be in the same order as defined in that block.

Number of Parameters = Number of parameters defined by this block. See note above concerning the Number of Scalers.

Scaler Value = The numerical value of the scaler.

Parameter Value (REAL*4) = The numerical value of the parameter.

Pointers	Description
SPVB Start $\longrightarrow$	Number of Groups = N_g
	Number of Scalers$_1$ = N_s
	Number of Parameters$_1$ = N_p
	Scaler Value$_{1,1}$
	...
	Scaler Value$_{1,N_s}$
	Parameter Value$_{1,1}$
	...
	Parameter Value$_{1,N_p}$
	Number of Scalers$_2$ = N_s
	Number of Parameters$_2$ = N_p
	Scaler Value$_{2,1}$
	...
SPVB End $\longrightarrow$	Parameter Value$_{N_g,N_p}$

II.D. Run Trailer

The run trailer is the last data for any complete run. Primarily, it contains run statistics.

Run Trailer = Trailer Flag
+ Trailer Length
+ Event Count
+ Date
+ Time
+ Run Status
+ Event File Locator List
+ Scaler/Parameter Value Block

Trailer Flag = A single word used to identify this as a Run Trailer. The standard Trailer Flag = −3333.

Trailer Length = Length (in words) of this event trailer block.

Event Count = The number of events in the event prior to this point.

Date = Current Date (see Run Header).

Time = Current Time (see Run Header).

Run Status = An integer value indicating the perceived status of the run. The current standard is that 0 implies good data, 1 implies the data is suspect for some reason and 2 implies the data is definately not useable.

Run Trailer Format Summary

Pointer	Description
Run Trailer Start $\longrightarrow$	Run Trailer Flag ($= -3333$)
	Trailer Length
	Event Count ($= 0$)
	Date_1
	Date_2
	Time_1
	Time_2
	Run Status
EFLL Start $\longrightarrow$	Event File Count $= N_{EF}$
	$\cdots$ (see Event File Locator List)
EFLL End $\longrightarrow$	Event File Name $_{N_{EF},8}$
SPVB Start $\longrightarrow$	Number of Groups $= N_g$
	$\cdots$ (see Scaler/Param. Value Block)
SPVB End $\longrightarrow$	Param. Value_{N_g,N_p}

Run Trailer Bank Structure

The run trailers are contained in a single bank structure. The additional CEBAF event format data is defined below.

Pointer	Word	Description
Bank Start $\longrightarrow$	1	Bank Length - 1
	2	Bank Tag (= 'CEBD1001'X)
Trailer Data Start $\longrightarrow$	3	Trailer Length - 1 (= Bank Length - 3)
	4	Trailer Tag (= '800B01000'X)
	5	Trailer Data
	6 $\cdots$	$\cdots$
Trailer Data End $\longrightarrow$	Bank Length	Trailer Data End

II.E. Converting Real and Character Data to Integer Data

When real values are stored in the event buffer they are placed in the buffer by equivalencing them to integers. For example:

```
program sample_convert
implicit none
real*4 real_value
integer*4 evgen_buffer(16384)
integer*4 swgen_r_to_i
evgen_buffer(1) = swgen_r_to_i( real_value )
stop
end


integer function swgen_r_to_i( real_value )
integer i
real x, real_value
equivalence (i,x)
x = real_value
swgen_r_to_i = i
return
end
```

Coversion of characters to integers is done by converting them character by character to integer bytes and then packing the bytes into integers. The first character goes into the first byte, the second character goes into the second byte and so on. If the character string does not evenly fill an integer, the integer should be filled with blanks which have been converted to integers. Any full integers which had been set aside for a character string should be set to zero. This is not the recommended CEBAF standard form for character strings, which should be null terminated and as a result wastes space. Again this will be corrected in the next version. For the present time, the conversion to integers before output, though wasting space and time for packing and unpacking, follows the standard form for integers.

IV. Summary and Future Directions

This document has presented all of the information necessary to write out CLAS event data files. The format presented here is expected to be the first version in a series of versions, all of which will conform to CEBAF's standard event format. Several of the items expected to be modified have been listed. Other items include those which are redundant. In those the items within the bank headers will be kept while the redundant items within the banks will be dropped. Those writing software should keep this in mind and use the information contained in the bank header whenever possible. Finally, the rest of the changes will optimize the event size by using smaller word sizes for experimental quantities and using the bank structure to organize the sectors, packages, subpackages and components.