LEGIBILITY NOTICE

A major purpose of the Technical Information Center is to provide the broadest dissemination possible of information contained in DOE's Research and Development Reports to business, industry, the academic community, and federal, state and local governments.

Although a small portion of this report is not reproducible, it is being made available to expedite the availability of information on the research discussed herein.

LA-UR-89-3596

LA-UR--89-3596

DE90 002410

Received by 077

NOV 0 6 1989

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7409-ENG-36

TITLE THE DEVELOPMENT OF A STANDARD FORMAT FOR
ACCELERATOR DATA ANALYSIS

AUTHOR(S) Stanley Cohen

SUBMITTED TO International Conference on Accelerator and Large
Experimental Physics Control Systems
Vancouver, British Columbia, CANADA

October 30 - November 3, 1989

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This report contains information that may be exempt from public release under the provisions of the Freedom of Information Act (5 U.S.C. 552). It is the policy of the Department of Energy to make this information available to the public as soon as possible after its release to the public.

This report contains information that may be exempt from public release under the provisions of the Freedom of Information Act (5 U.S.C. 552). It is the policy of the Department of Energy to make this information available to the public as soon as possible after its release to the public.

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

THE DEVELOPMENT OF A STANDARD FORMAT FOR ACCELERATOR DATA ANALYSIS *

Stanley Cohen
Los Alamos National Laboratory
MS H852
Los Alamos, NM 87545 USA

Abstract

The purpose of specifying a standard file format is to facilitate the analysis of data sampled by accelerator beam diagnostic instrumentation. The format's design needs to be flexible enough to allow storage of information from disparate diagnostic devices placed in the beam line. The goal of this project was to establish a standard file layout and syntax that can be generated and "understood" by a large set of applications running on the control and data-analysis computers at LAMPF as well as applications on personal computers. Only one file-parsing algorithm is needed for all computing systems. It is a straightforward process to code a parser for both the control computer and pc's once a consensus on the file syntax has been established.

This paper describes the file format and the method used to integrate the format into existing diagnostic and control software.

Introduction

The motivation for specifying a standard storage format for accelerator data analysis is to allow as many applications as possible to share information easily. The proliferation of personal computers and workstations has created an additional demand for files that are transportable across machine boundaries. One of the goals of this project was to specify a file format syntax that requires little or no intermediate processing between applications.

Data acquisition programs written by the controls group at the Clinton P. Anderson Meson Physics Facility (LAMPF) have always

* Work supported by the U.S. Department of Energy

saved data to disk. Each file format, however, was different. In some cases they were stored in binary form. Applications could not easily exchange information.

Many people doing accelerator development are now using commercial data-analysis tools. These applications expect a specific file syntax for their input data. The files created by the suite of file-format subroutines that are described below are understood by most of the popular commercial analysis and plotting packages that run on MS-DOS and Macintosh personal computers. The files are coded in ASCII¹ so that no special protocols are needed to transfer the files between machines. The files can be printed easily.

The retrofit of the new file-storage mechanism into existing programs had to be done with a minimum amount of effort and, most importantly, not induce any undesirable side-effects.

The file syntax.

The file syntax chosen is straightforward and simple. The specific design allows different pieces of software to read the file. The standard file-format syntax rules are:

Rule #1. The file character set consists of ASCII characters.

Rule #2. The file is made up of a series of lines terminated with a carriage return.

Rule #3. Each line can contain any number of space delimited and/or tab delimited fields or be a "comment line"

A "comment line" is any line whose first non whitespace character is an exclamation point "!" or asterisk "*".

Whitespace is interpreted by the parser as a space (ASCII 32), a tab (ASCII 9), line feed character (ASCII 10), a line feed character (ASCII 12) or a carriage return (ASCII 13).

Rule #4. Rule #3 effectively limits the characters used in the fields to characters of ASCII 33 or greater. In other words, the fields cannot contain imbedded blanks or whitespace.

An installation example

One popular diagnostic tool for tuning the proton storage ring (PSR) is the beam-position-monitor² (BPM) program. There are two general categories of information that developers need from this program.

1. A "snap-shot" of the values or states of selected device channels of the accelerator at the time the beam position data was sampled. This is appropriately called accelerator-state data.
2. The output from a selected set of monitors that are placed in the beam line. Each BPM records the instantaneous transverse horizontal and vertical position of the beam as well as its relative intensity.

To accumulate accelerator-state data, a user-generated input file is required. It specifies names of the channels of interest. These channels can refer to any device that is contained in the accelerator control system database. The accelerator-state file is an ASCII file that adheres to the syntax rules stated above.

This accelerator-state file is, in fact, the simplest example of the standard file syntax. Each line in the file contains either a comment or single-device channel-name field consisting of a maximum of ten characters. Characters after the tenth are ignored by the accelerator state parser.

A typical accelerator state file is illustrated in Figure 1. The values of the channels contained in the accelerator state file are included in the BPM output file. The standard file format routines produce the output file comprised of three sections.

1. Date and time that this file was created
The name of the accelerator state file (input file) and the name of the file created (outputfile) are included in this section.
2. List of channels and values in engineering units

Any channel that cannot be evaluated is converted into a comment line and labeled as an invalid channel name.

3. Application-dependent data -

In general, the data for each diagnostic detector of interest is multi-dimensional. It is assumed that each line will be devoted to a single detector. This is illustrated by the output file from the BPM diagnostic program in Figure 2.

The user interface of the beam-position-monitor program is a touch-sensitive screen. It is rather laborious to "type" file names using the touch-screen keyboard. In order to specify the name of the accelerator-state file and the name of the output data file, a third file, SAVE_BPM.FILES, is employed. It contains the full path names of the files to be used by the BPM application, and can be edited at will so that different accelerator-state files can be used. In other words, different accelerator-state files can be created and saved. All that one need do is change the name of the accelerator-state file in SAVE_BPM.FILES so that it "points" to the desired file. At almost any time, SAVE_BPM.FILES can be edited and a different state file can be referenced.

The second line of SAVE_BPM.FILES is the default output file name. In practice, the save-file module, of the BPM program generates a file name based upon the system time. The default file name is used if the access to the date_time system service is disrupted in some way.

The last line of SAVE_BPM.FILES contains the name of the field delimiter used in the output file. At this point there are two choices, "tab" or "space". A typical SAVE_BPM.FILES is illustrated in Figure 3. Commercial personal computer programs such as Lotus 1-2-3³ and Microsoft Excel⁴ will automatically parse tab delimited data fields.

Utility Programs

Both utility programs are installed as VAX DCL⁵ commands. They are invoked by merely typing the program name at the system

prompt. Both expect a file pathname as an argument. If one is not supplied on the command line, the user will be prompted for it.

These two applications are illustrative of the use of the line-parsing routines contained in the fileform library. Users may include the capability of reading files in the standard format in their own applications. For example, one can easily use the library to write a routine that accepts the name of an input file and a field position in a line. An array of all of the values in that field position can be returned to the calling program.

The utility, CHECK_FORM, evaluates the file specified for syntax. The output of this check is written to the file CHECK_FORM.CHK. CHECK_FORM performs four basic checks. As each line is read a parser decides whether or not it is a comment line. Comment lines are labeled as such, assigned a line number, and passed to the output file. All other lines are assumed to be valid data. Each line is assigned a line number. All embedded whitespace is replaced with the circumflex character (^). The number of fields that are contained in the line is calculated. The number of characters in a line is also reported. Leading whitespace is stripped.

The result of evaluating the file in Figure 2 with CHECK_FORM is shown in Figure 4.

After creating a new accelerator state file, it is useful to run VERIFY_CHANNEL to make sure that all of the channels are actually accessible. VERIFY_CHANNEL produces an output file called VERIFY_CHANNEL.CHK. The output file is shown in Figure 5.

Standard file-format-module design and installation

A basic requirement of the standard file format project was that it be easy to retrofit the file generating mechanism into existing applications. The set of file format subroutines is designed in such a way that in principle only one call need be added to existing source code. There is only one subroutine that the file format package needs that must be rewritten for each application. The suite of routines was put in an object library. One application specific routine, described below, is compiled as a separate module.

The only call that is made in the application program is to fileform. Fileform requires three character strings as arguments:

SUBROUTINE FILEFORM(INFILENAME, OUTFILENAME, CHAR_NAME).

1. **INFILENAME**, the pathname of the accelerator-state file.
2. **OUTFILENAME**, the pathname of the output data file.
3. **CHAR_NAME**, the name of the field delimiter character, such as "space" or "tab".

Subroutines in fileform take care of all of the file i/o housekeeping chores, interpreting device channels specified in the accelerator-state file, fetching the values of the channels and handling any errors.

Fileform references one routine that is not in its library called **LOCAL_DATA_WRITE**. This is the routine that must be written specifically for each application. Local_data_write is passed two values from fileform, the logical unit number of the output file and the delimiter character to be used. In practice, this application-specific routine needs to have access to the data accumulated from its host program. In the case of the diagnostic programs for the PSR the data is invariably stored in a set of arrays stored in a common block. It is a straightforward programming task to access the arrays and write them to the disk file. Each value is separated by the delimiter character. The interrelationship between the different parts of the file format modules is illustrated in Figure 6.

Conclusion

The standard file format has been in use for about four months. The retrofits to two existing programs went quite smoothly. The second installation was done by another programmer who knew nothing about the internal structure of fileform, but who had the file saving function working within a day or two. The beam line developers have been pleased with the ease with which the output files can be imported to applications on their personal computers.

References:

-
- ¹American Standard Code for Information Interchange
 - ² E. F. Higgins and F. D. Wells, "A Beam Position Monitor System for the Proton Storage Ring at LAMPF", IEEE Trans. Nucl. Sci. **28** (3) 2308-2310, June 1981
 - ³ Lotus 123 Spreadsheet program for MS-DOS. Lotus Development Corporataion, Cambridge MA.
 - ⁴ Excel Spreadsheet program for Apple Macintosh and MS-DOS computers Microsoft Corporation, Redmond, WA
 - ⁵See Digital Equipment Corporation VAX manual, "Guide to Using DCL and Comand Procedures on VAX/VMS".

Figure Captions:

Figure 1. The accelerator-state file (an input file)

Figure 2. Data output from the BPM application

Figure 3. A typical SAVE_BPM.FILES the first line is the user-created accelerator-state file. The first line specifies the input accelerator-state file, the second line is the default output file. The last line is the name of the delimiter to be used to separate the data fields of the output file.

Figure 4. A typical output of CHECK_FORM. Each line is considered either a comment line or data line. The numbers in the left columns separated by the vertical line (|) show the line number, the number of fields that have been parsed and the number of characters in the line, respectively. Each circumflex (^) replaces a space contained in the line.

Figure 5. VERIFY_CHANNEL output file.

Figure 6. An illustration of the interrelationship of the parts of the file form library and the application in which is installed. In its minimum configuration, only one subroutine call need be inserted into the main application code. Any other processing can occur in a separate module. The routine LOCAL_DATA_WRITE typically obtains data values through globally defined arrays.

```
RIKI01-14
RIKI01-11
RIKI01-06
RIKI01-03
RICM01-02
LDCM03-02
! I know that the next device doesn't exist
LRCM04-05
RICM02-02
RICM03-02
RIKI01-15
RITT01-02
RITT001L06
66TUL1
CCBG003L01
RODM002D04
RITT01-09
RITT01-10
1DTI01-02
```

Figure 1

```

* 31-JUL-1989 10:38:36.65
! Input file : control_beam_pos_mon:acc_state.list
! Output file : control_beam_pos_mon:JUL_31_1038.BPM
* Accelerator State Section
! This is a comment line
RIKI01-14      0.176E+03  COUNT
RIKI01-11      0.175E+03  AMPS
RIKI01-06              ON
RIKI01-03              ON
RICM01-02     -0.615E+02  uAMP
LDCM03-02     -0.614E+02  uAMP
! I know that the next device channel doesn't exist.
! LRCM04-05  Invalid channel name
RICM02-02     -0.615E+02  uAMP
RICM03-02      0.590E+01  uAMP
RIKI01-15     -0.693E-02  PCENT
RITT01-02      0.220E+02  mm
RITT001L06              NOT
66TU1E1       0.879E+03  CNTS
CCBG003L01              NO
RODM002D04     0.870E+01  AMPS
RITT01-09              NOT_LIMIT
RITT01-10              NOT_LIMIT
1DTI01-02      0.318E+03  DEGC
* Data Section
! Device      H_pos(cm)  V_pos(cm)      Intensity
RI-01         -0.082     0.176      0.270E+02
RI-02          0.223      0.000      0.290E+02
RI-03          0.100     -0.067      0.150E+02
RI-04         -0.211     -0.012      0.310E+02
RI-05          0.000     -0.081      0.270E+02
RI-06          0.047     -0.093      0.350E+02
RI-07          0.432     -0.012      0.350E+02

```

Figure 2

```
control_beam_pos_mon:acc_state.list  
control_beam_pos_mon:bpm_anal.dat  
space
```

Figure 3

```

* 3-AUG-1989 08:26:08.45
! check form input : JUL 31 1038.BPM
1| COMMENT| * 31-JUL-1989 10:38:36.65
2| COMMENT| ! Input file : control_beam_pos_mon:acc_state.list
3| COMMENT| ! Output file : control_beam_pos_mon:JUL_31 1038.BPM
4| COMMENT| * Accelerator State Section
5| COMMENT| ! This is a comment line
6| 3| 29| RIKI01-14^00000.176E+03^COUNT
7| 3| 28| RIKI01-11^00000.175E+03^AMPS
8| 2| 18| RIKI01-06^0000000ON
9| 2| 18| RIKI01-03^0000000ON
10| 3| 28| RICM01-02^000-0.615E+02^uAMP
11| 3| 28| IJCM03-02^000-0.614E+02^uAMP
12| COMMENT| ! I know that the next device channel doesn't exist.
13| COMMENT| ! IJCM04-05 Invalid channel name
14| 3| 28| RICM02-02^000-0.615E+02^uAMP
15| 3| 28| RICM03-02^00000.590E+01^uAMP
16| 3| 29| RIKI01-15^000-0.693E-02^PCENT
17| 3| 26| RITTO1-02^00000.220E+02^nm
18| 2| 19| RITTO011.06^0000000NOT
19| 3| 28| GGTU1E1^000000.870E+01^CNTS
20| 2| 18| CUBG00 4.01^0000000NOT
21| 3| 28| FOMMO0T04^000.870E+01^AMPS
22| 2| 25| RITTO1-09^0000000NOT LIMIT
23| 2| 25| RITTO1-10^0000000NOT LIMIT
24| 3| 28| IJPT01-02^00000.319E+01^DEGR
25| COMMENT| * Data Section
26| COMMENT| ! Device H pos(cm) V pos(cm) Intensity
27| 4| 44| FI 01^0000000-0.082^00000.176^00000.270E+02
28| 4| 44| FI 02^0000000-0.22^000000.000^00000.290E+02
29| 4| 44| FI 03^0000000-0.100^00000.067^00000.150E+02
30| 4| 44| FI 04^0000000-0.211^00000.012^00000.310E+02
31| 4| 44| FI 05^0000000-0.000^00000.081^00000.270E+02
32| 4| 44| FI 06^0000000-0.047^00000.094^00000.290E+02
33| 4| 44| FI 07^0000000-0.43^00000.012^00000.310E+02

```

Figure 4

```

! Input file : [cohen.pfiles]acc_state.list
! Output file : [cohen.pfiles]verify_channel.chk
* Accelerator State Section
! This is a comment line
RIK101-14      0.168E+03  COUNT
RIK101-11      0.168E+03  AMPS
RIK101-06      ON
RIK101-03      ON
RICM01-02     -0.646E+02  uAMP
LRCM03-02     -0.650E+02  uAMP
! I know that the next device does not exist
! LRCM04-05  Invalid channel name
RICM02-02     -0.645E+02  uAMP
RICM03-02      0.501E+01  uAMP
RIK101-15     -0.944E-02  PCENT
RITT01-02      0.220E+02  mm
RITT0011.06   NOT
CCT011E1      0.910E+03  CNTS
CCT0031.01    NO
R04M0021.04   0.872E+01  AMPS
RITT01-09     NOT LIMIT
RITT01-10     NOT LIMIT
RITT01-07     0.404E+03  DEGR
* Data Section
! This is a channel check output file -- no data

```

Figure 3

Figure 6