File: IBM Formatted File System Rendering 3-5-2.doc
Author: Kenneth Grant
Status: Version 3.4.1, Published.
Created: Wednesday, October 10, 2007
Updated: Tuesday, October 14, 2008

IBM Formatted File System (FFS) to ASCII Tables for NIPS files Rendering Engine User Manual

1. Summary.

This document details the functioning and use of NIPSTRAN, a program for rendering NIPS database files into ASCII tables for use with ASCII character set supporting computers. The files used as input are IBM Formatted File System (FFS) files. These NIPS files, dumped in image form, are typically from NMCS, the National Military Command and Control System's Command and Control Technical Centers. They span the years from mid 1960 to early 1980's. These files are referred to as NIPS (NMCS Information Processing System) files and generally contain military information about actions during the Vietnam War. However, there exist files of non-military data such as those found in the Nixon Library.

2. History

To be supplied at a later date if resources are available.

3. Overview of NIPS Files

NIPS files are nothing more than IBM Formatted File System (FFS) files with a few additional built-in data types. Reporting features (not implemented by this program) also have additional features to allow display of these additional data types. A more detailed description of system concepts can be found in NMCS Information Processing System FFS Users Manual [1]. The following is a much shortened summary of this document's description.

It is important to understand that NIPS FFS was an implementation of very early efforts in database design. In some ways, it represented very rudimentary relational database concepts. Terminology, however, was inconsistent, with "sets" having multiple meanings. Nowhere does one find terms like "tables" or "data elements" although those are exactly the data concepts being described by the NIPS documentation.

In this description, I will attempt to use the modern terms. When reading the historical documentation, the readers may indulge themselves in the antiquated terms or use the terms in this overview.

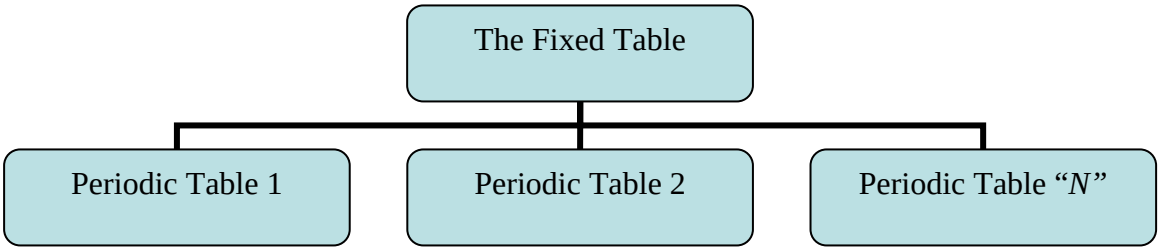
A NIPS database consists of data elements organized into "Sets". The current view of these sets would be tables as all records, or rows, in these sets have the same format. Unfortunately, the historical documentation never refers to individual rows in these tables as records. The meaning of records in the historical documentation was logical assemblies of various rows from all the sets. Regardless of the terminology, the results are the same. So let us begin.

A NIPS database has one table (referred to as the Fixed Set) and multiple additional tables (referred to a Periodic Sets). There is a unique "key" in each row of the Fixed set which points to none to many rows in each Periodic Sets. The key is called the Control Set (an ambiguity since "Set" now has multiple meanings, I.E. a table and a

key).This represents a one-to-many relationship and provides a hieratical relationship of one and only one level.

Each row in a periodic set has none-to-many occurrences of the Control Set (key). A Subset (yet another use of the word set) is appended to the Control set (key) and represents a unique row in each periodic set (Table). I think that the reader can see how confusing this historical terminology can be. Let us try and restate this in relational database terminology.

A NIPS dataset has a base table which logically points to one or more hieratically related tables in one to many relationships. This is shown as:



There is a concept in NIPS that periodic tables can have horizontal relationships in the above diagram but there is no clear way that this can be described in the metadata of the IBM Formatted File System. Unless the documentation clearly describes this relationship, it has to be left to the researcher's investigation of the data to determine.

Previous translations by NARA of some of these files attempted to utilize the concept described above. This was done by defining a record as the amalgamation of the record layout for the fixed set with the record layout of each periodic set. For example, a record layout was defined as:

FT Layout + PT 1 Layout + PT 2 Layout + … + PT N Layout

Where: FT is the Fixed Table Row Layout,
PT 1 is Periodic Table 1 Row Layout,
…
PT *N* is the *N*th Periodic Table.

If there were no more entries in one of the periodic tables, blank space would simply be supplied for those fields in this amalgamated record. This model was consistent with the described design of the FFS but caused both expansion of the data and a possible misinterpretation of the data if that model was not the intent of the database designer. The researcher is better served in using a relational model of individual tables as he may use modern tools and assemble that data as deemed fit.

This relational model was recognized by W. I. of the MITRE Corporation in 1972[(2)] as relational database concepts evolved. As a result, many, if not most, NIPS files were really relational in structure, not flat across all tables.

4. Program Features

The program, NIPSTRAN, developed by NARA, Electronic Records and Special Media Division, provides a tool to both analyze and extract into relations tables the data content of NIPS System files that were copied and transferred to NARA in image form. Hundreds of these files have been persevered by NARA. Many of these files were transferred to NARA without any or little documentation. For several decades, this has been a serious problem as NARA has had no way to accession these image files. With this additional tool, the metadata stored in these image files can be extracted and used to build relational tables. What remains as the more challenging task is the intellectual interpretation of the column labels found in the metadata. Since the NIPS system only allowed seven uppercase letters and numbers for labels, the set of possible labels is somewhat constricted. It was believed that many labels would be re-used between databases for similar data and as a result, the concept of a data dictionary, to be used by the program was introduced. An unexpected benefit to using the Data Dictionary was to capture agency documentation in an electronic form, matching that documentation with what was actually found in the image files.

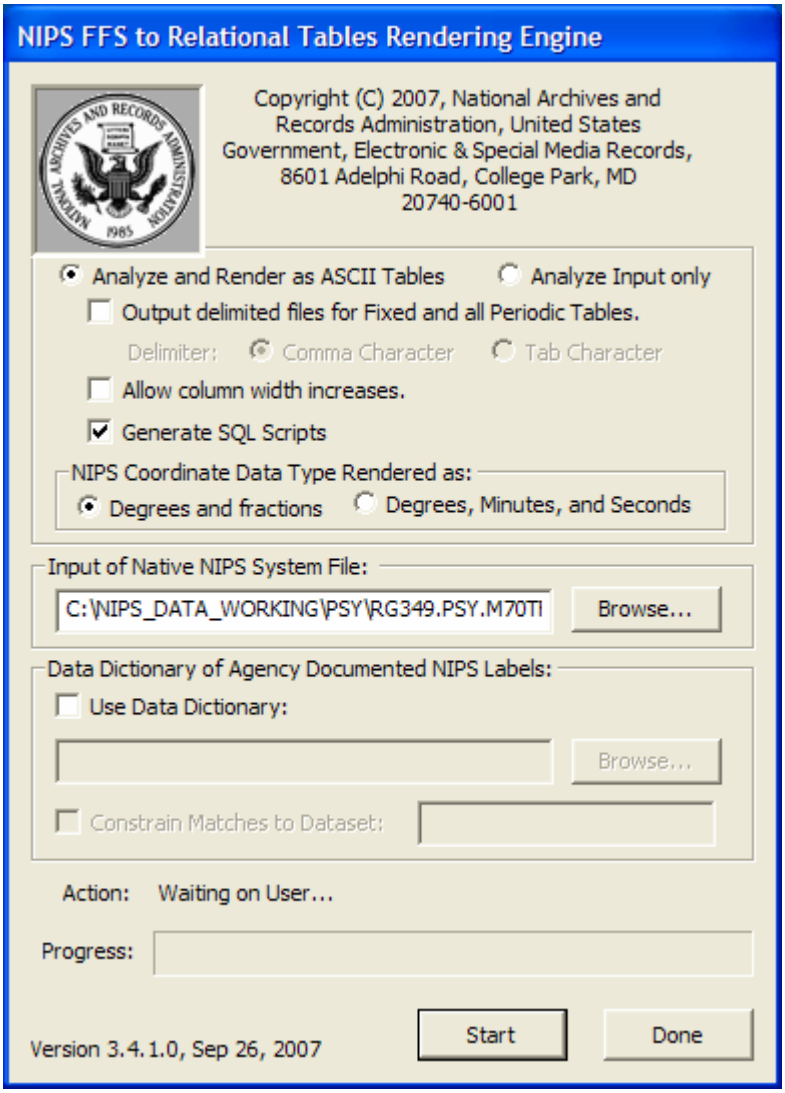
Since this project was not funded, beyond using time-available from a NARA staff member, a simple use of an Excel spreadsheet provided the user interface for a data dictionary. In order for the program to utilize the content of the data dictionary, its contents are exported into a comma separated values, or CSV, file, which the program can easily read.

When a NIPS image file is read by the NIPSTRAN program, each discovered label is looked up in the data dictionary and any matches found have those descriptions printed along with the discovered layout for each table in the database. By adding entries in the data dictionary, archivists may build a cross reference of label uses that hopefully help in both keeping track of the meaning and enhance the archivist efforts to determine the meaning of undocumented labels.

Another use of the Data Dictionary would be to key into separate spread sheet the agency documentation for the data element or label used buy the NIPS file. Output from NIPSTRAN would then provide a detailed description of each data element in a generated report.

5. Program User Interface

The program requires no installation as system wide settings are not required to run the program. A user or archivist simply copies the program executable onto his Windows Workstation Desktop. The program requires Windows XP or later. Below is the User Interface to the program as it appears during its use.

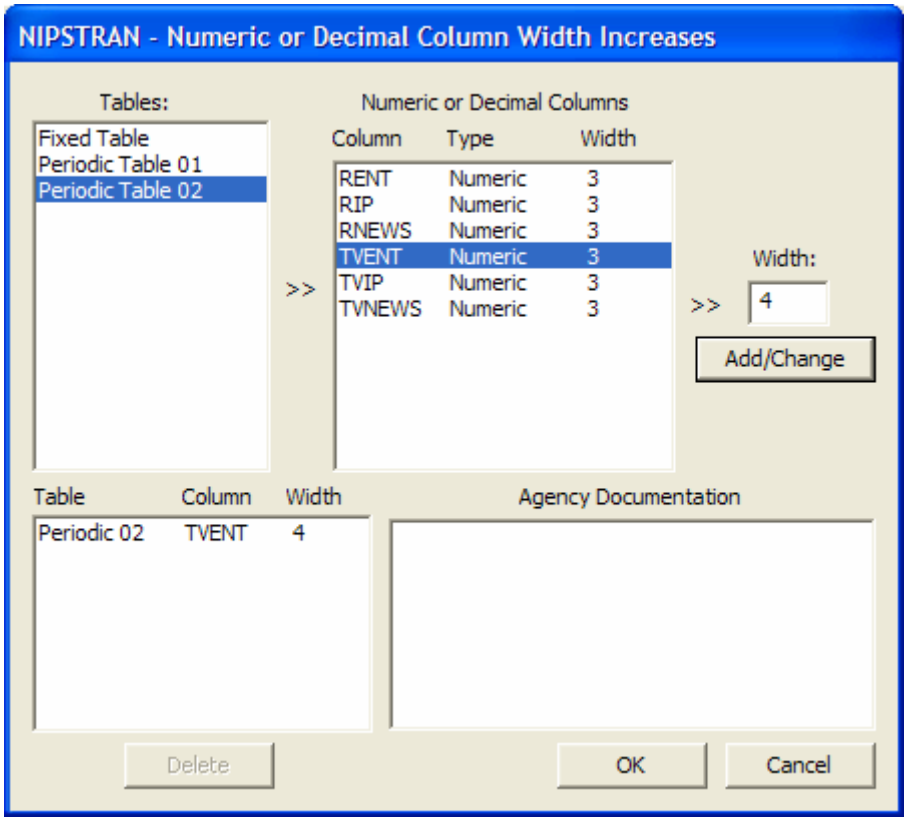


Two action choices are provided. A NIPS file can just be analyzed or it can be analyzed and rendered into its table components. The default is to analyze and render.

If being rendered, in addition to rendering into a fixed field sized data records (each terminated by a carrier-return line-feed ASICC character) it may also be rendered into files utilizing the industry standard delimiters. The two choices for delimiters provided are commas and tabs. The comma separated values choice is the default. This allows modern tools like Access™ and Excel™ to load the tables for researcher

interpretation. Should some fields of data contain commas (although unlikely, it could happen in perhaps a remark field) the tab option use of the "tab" character would eliminate a conflict between field separators and data content.

Another choice is to allow for management of column widths. This is necessitated by discovery of NIPS files that contain data for which the width defined by the metadata is insufficient. For example suppose a width of two was specified for a column like "Weapons Found" but the value of 120 was stored in the NIPS file. In order for all the fixed length records to have the same length, it is necessary to increase the column with above that specified in the metadata. If the user selects this option, then immediately after reading the metadata, the program present dialog to allow changes in the widths. Below is an example of this screen.



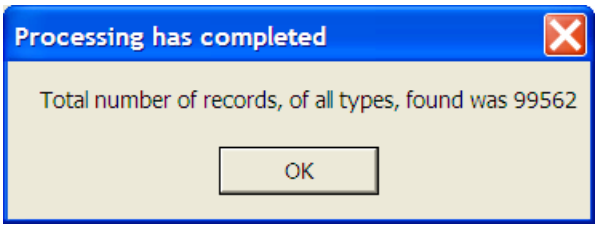
Note that Agency Documentation, if present, is shown for the selected data element. Use of the dialog follows use in many Windows applications and, hopefully, should be self explanatory.

If the column width change option is chosen, an additional file is created to "remember" the changes in any future execution of the program. Although this file is in ASCII, hand edits of this file should not be attempted. The file has the same base name as the file being processed with "WIDTH" as the file type.

The last option before the data dictionary is the generation of SQL scripts to load a relational database with the rendered NIPS file. These SQL scripts have been tested with current (2007) versions of both Oracle™ and MySQL. Comments in the script show where users can, if they wish, change location specifics for source table data.

The use of a data dictionary is optional. If used, the user can "constrain" the matches to a single group of data files or a single file through the use of a grouping name stored in the data dictionary.

Since the processing of even large NIPS file containing tens of thousand of records only takes fractions of a minute, the program does not implement a cancel button. Once execution of the program completes, the total number of records found is displayed in a message dialog box as shown below. The count includes all records, regardless of record types.



6. Input Files

6.1. NIPS Files

The input to the program consists of an IBM FFS file dumped in image form. This file must be unaltered from the original which includes the record lengths as the first data in each record. Unfortunately, it has been discovered that some files were either transferred to NARA in a different format or were later altered before preservation. Additionally, the Archival Preservation System (APS) can produce reference copies using various options that make the resulting file unusable by this program. The archivist must be sure the reference copy produced by APS is a "Smart" copy.

6.2. Data Dictionary File

The data dictionary file read by the program is the comma separated value file exported from an Excel™ spread sheet. It is *very* important that the data dictionary are maintained and edited using Excel. Do not edit the data dictionary using the tab exported file. There are two sections to the data dictionary. The first section is for defining tables that are output. The second is for elements within a table. Within each section, the first row of a CVS file contains the column labels and is ignored by the program. All additional rows are defined by the two tables below. Each section is bracketed with a section header and a section end as shown in the example below.

Note: The excerpt below uses commas for clarity but tabs are used in the actual file.

<Table Definitions>
Table, In Dataset, -,-,-,-,Name,Description, Source, Authority
… Column data for table definitions is located here
<End>

<Data Definitions>
Mnemonic, In Dataset, Type Mnemonic,Length Characters,Code TableType,…
… Column data for data elements definitions is located here
<End>

For the Table Definitions section, each additional row has the following layout:
(Note: each column is separated by a tab so tabs must not be used by the archivist in entering data into the spread sheet. A warning message will be printed into the report and the line ignored by the program if there is an extra or missing tab.)

Column	Label	Use
A	Table Number	This is the table number being defined. “0” is used o represent the Fixed Table.
B	In Dataset	This column allows the archivist to collectively group tables for commonality in table interpretation.
C	-	This column is not used.
D	-	This column is not used.
E	-	This column is not used.
F	-	This column is not used.
G	Name	The name or title of the table. This is the text printed in the report by the NIPSTRAN program
H	Description	This is a longer description of the meaning of the use of the table in the file.
9 I	Source	This is the source of these details. I.E. “AMFESMA File Detail (Data Dictionary)”
J	Authority	This is the authority for these details. I.E. “Agency Documentation“

For the Data Definitions section, each additional row has the following layout: *(Note: each column is separated by a tab so tabs must not be used by the archivist in entering data into the spread sheet. A warning message will be printed into the report and the line ignored by the program if there is an extra or missing tab.)*

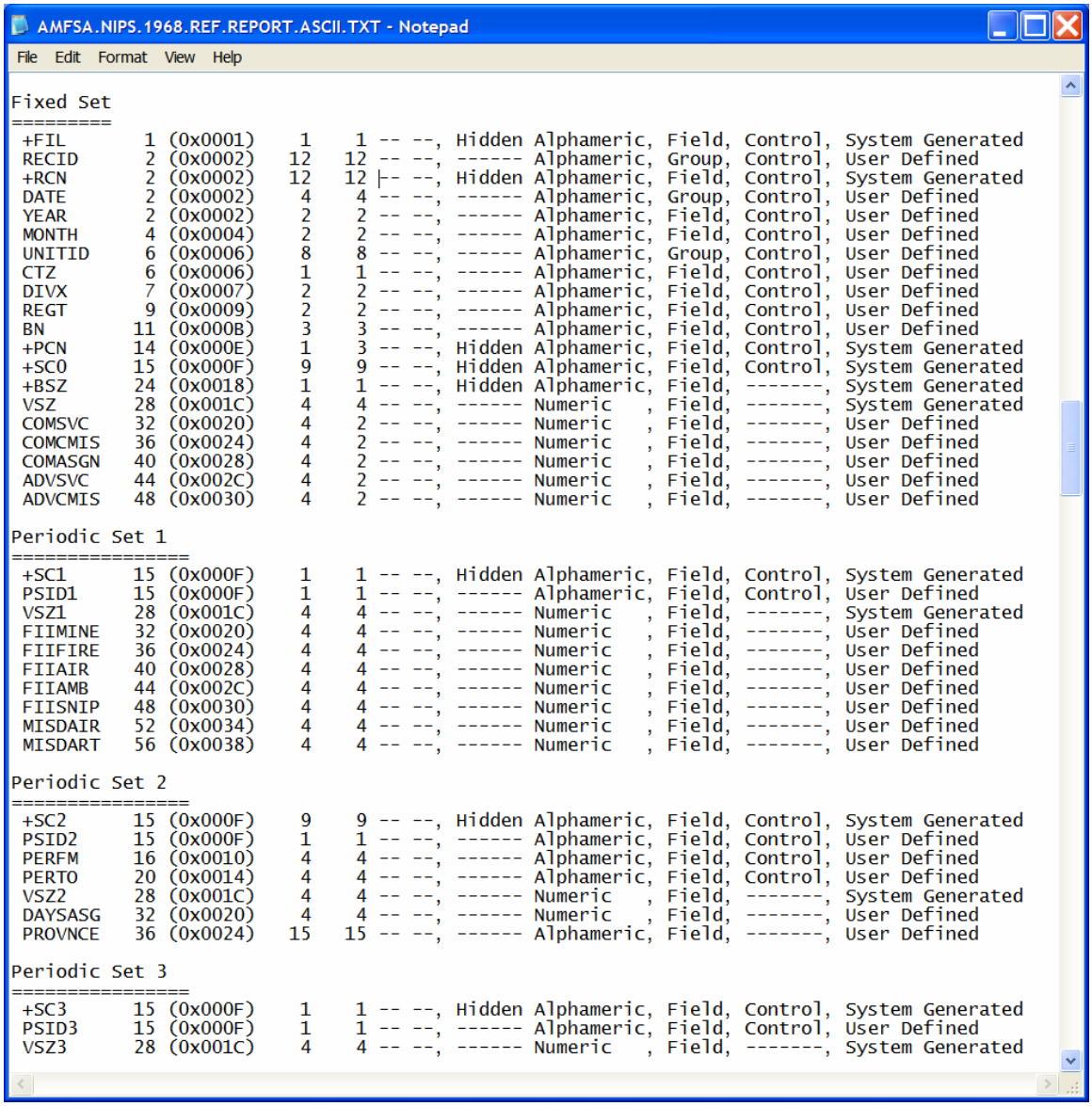
Column	Label	Use
A	Mnemonic	This is the 1 to 7 character label found in the metadata of the NIPS file.
B	In Dataset	This column allows the archivist to collectively group labels for commonality in label interpretation.
C	Type Mnemonic	This column indicates the label type. The only labels used by the program are those which have “Field” as the Type Mnemonic.
D	Length Characters	This is the size of the data field. It is important to note that during rendering of the data into tables, the rendered field size may change to accommodate the encoding to ASCII.

E	Code Table	If a field contains values, the name of the code table is shown in this column. The name is a tab in the Excel spread sheet for use by the archivist during accessioning.
F	Type Data	This is the type of data stored in a field. The values are Numeric, Alphameric, Coordinate, etc.
G	Name	The name or title of the label. This is the field printed in the report by the NIPSTRAN program
H	Description	This is a longer description of the meaning of the use of the label in the file.
I	Source	This is the source of these details. I.E. “AMFESMA File Detail (Data Dictionary)”
J	Authority	This is the authority for these details. I.E. “Agency Documentation“

7. Output Results

7.1. Analyze Only – Report File

When the user chooses “Analyze Only”, one report file is generated. This file has the same name as the input file with “.REPORT.ASCII.TXT” appended to the file name. This report file details facts found in the NIPS file and then generates the layout of the data content of this NIPS file. Below is an example of the layout section of the report file opened in Notepad.



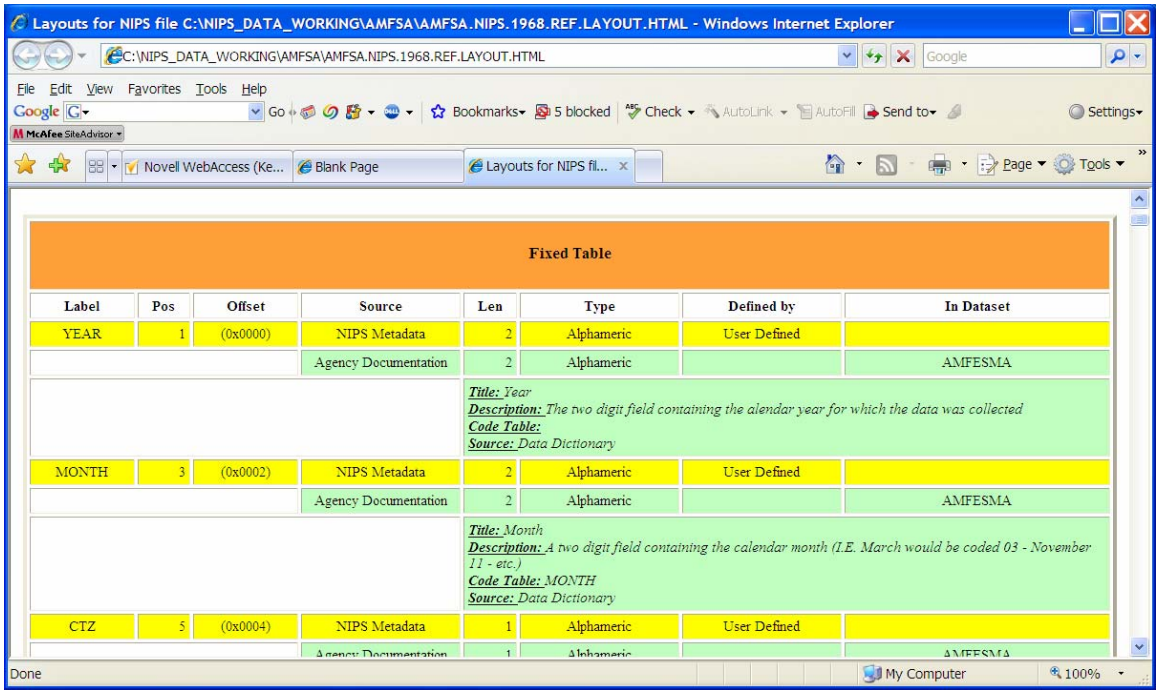
It is important to note that this is the input format of the data, not the rendered output. The columns in the report are label, offset in the file, both in decimal and hexadecimal, the stored or internal length, the external length of the field, two flag columns, an attribute of hidden, the data stored type, a indicator as to the labels status as a group of fields (a collection of labels that follow) or a single field, an indication of Control if part of the key, and who generated the label.

7.2. Analyze and Render – Layout File

If the user chooses the Analyze and Render option, then files additional to the report file are generated. These include a “Layout” file, two files for each “Set” or Table in the NIPS file, and two files for AERIC, a verification program used by NARA archivists.

The generated “layout” file has the same name as the input file with “.LAYOUT.HTML” appended to the file name. The file contains the generated layout

of each table. Below is an example of these layouts in the layout file opened in Notepad.

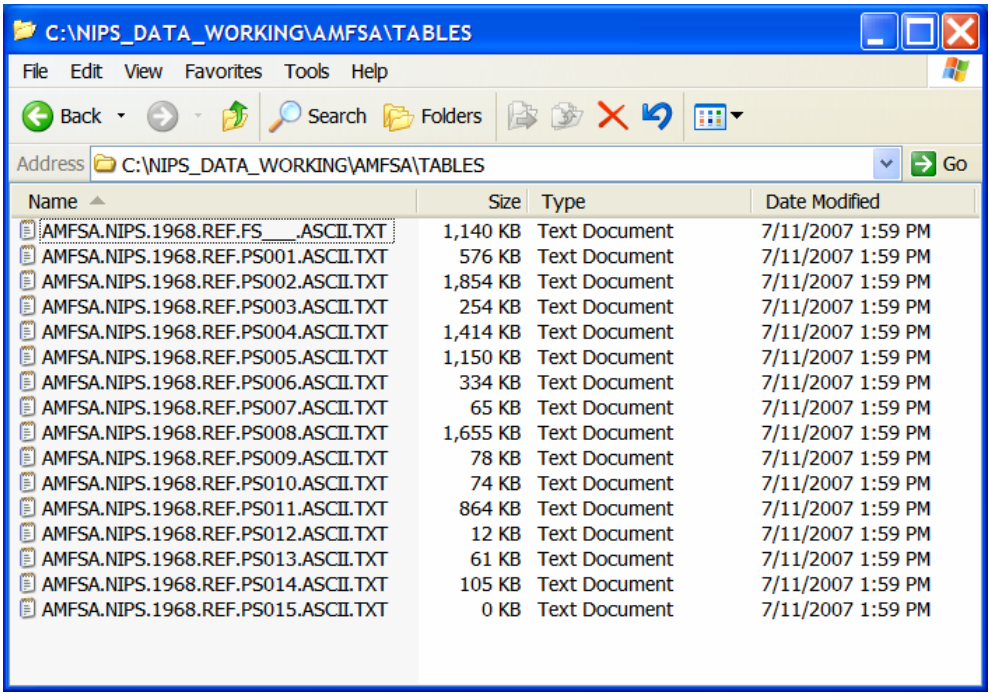


If you compare the two listings, you will immediately notice many differences. The input Report shows each record starting at offset 1 as opposed to the Layout which starts at zero. However, the position in the file starts at one since most researchers use a one based counting system. The next column identifies from where the information is being drawn. Then the next column is the length of the stored value. Since this is always ASCII, the stored length is always the external or display length. This is followed by the data type, who defined the label if the information was extracted from the NIPS file, and then the dataset grouping if from agency documentation.

The user should note that printing this report in color requires that the browser (I.E. Internet Explorer) have “Print background colors and images” checked. This is found in the “Tools” Menu, “Internet Options…”, “Advanced” tab, under “Printing.”

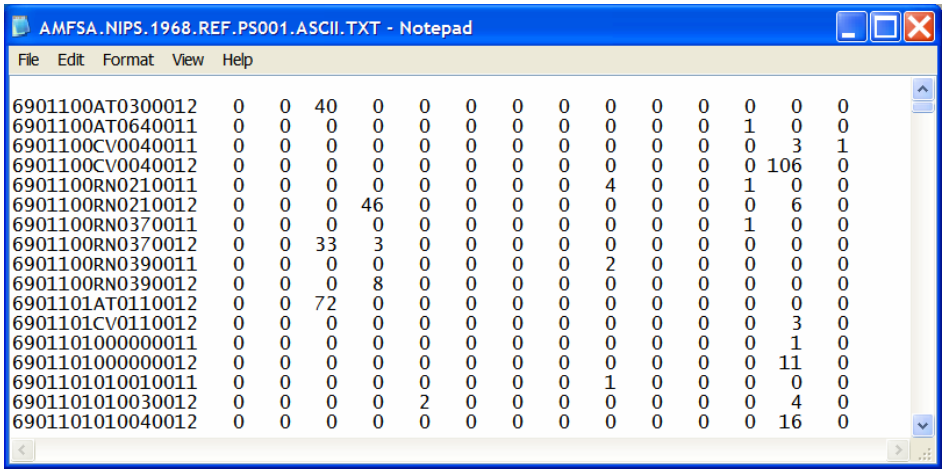
7.3. Table Output

When rendering output, the program always outputs fixed fielded records of constant length. Each table representing each “Set”, both the Fixed Set and the Periodic Sets, are output as separate files. All such table output is place in a folder or subdirectory with the name “TABLES”. The fixed set file name is formed from the input file name with “.FS___.ASCII.TXT” appended. Each Periodic Set file name is formed from the input file name with “.PSxxx.ASCII.TXT” appended where “xxx” is the Periodic Set number. Shown below is the directory listing of a rendered NIPS file, AMFSA.NIPS.1968.REF” with fifteen (15) Periodic Sets.



Many tools, such as Microsoft Excel, can import these files by allowing the user to designate the column locations for each field. However, column labels are not imported and are provided by NIPSTRANS in these fixed field records. Importing records from NIPS is more easily done using the delimited files.

Below is a sample of one of Periodic Set 1:

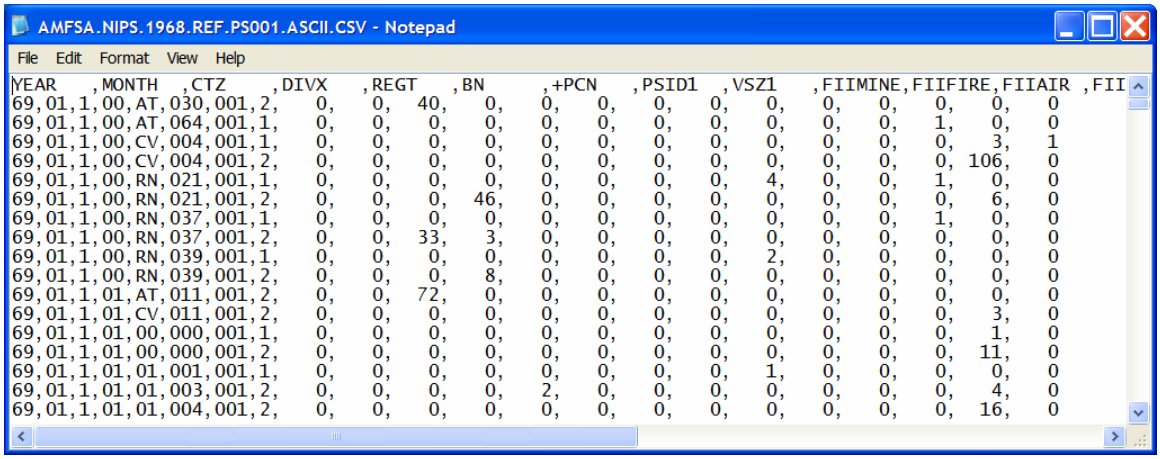


7.4. Delimitated Files

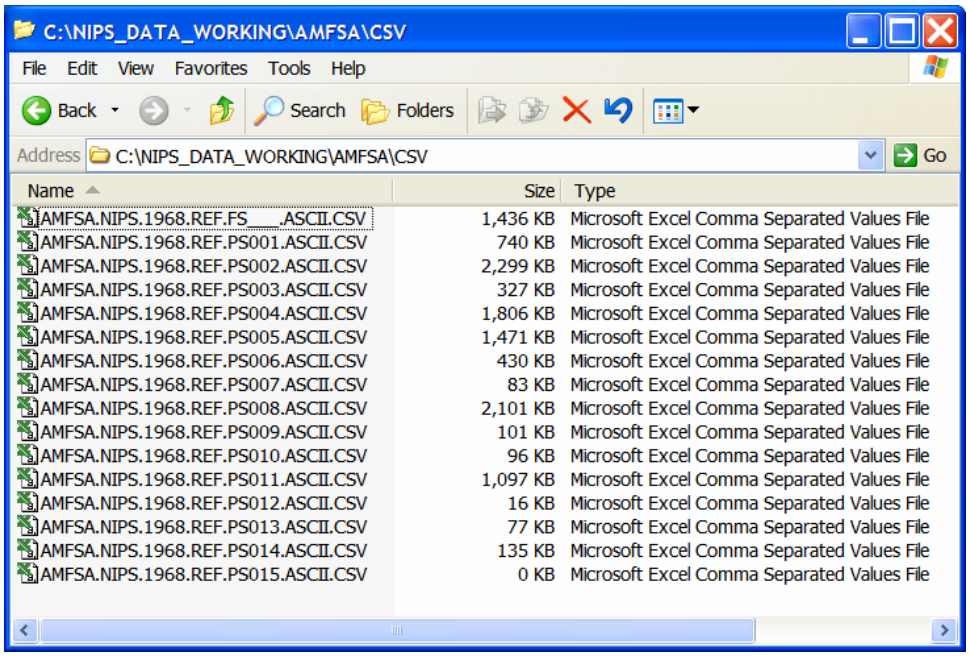
Delimited files have a delimiting character between each field. The default delimiter is a comma but the user may specify the use of a tab for the delimiter. Rendering output as delimited is optional. Each table representing each “set”, both the Fixed

Set and the Periodic Sets, are output as separate files. All such table output is placed in a folder or subdirectory with the name “DELIMITED”. The fixed set file name is formed from the input file name with “.FS___.ASCII.CSV” appended if comma delimited or “FS___.ASCII.TAB” if tab delimited. Each Periodic Set file name is formed from the input file name with “.PSxxx.ASCII.CSV” appended if comma delimited and “.PSxxx.ASCII.TAB” appended if tab delimited. (The designation “xxx” is the Periodic Set number.)

Shown below is a sample of the same Periodic Set 1 shown above. Notice that the first row includes the field labels. If a label is preceded with a special symbol, I.E. a plus sign, then the delimited output of the first row has a single blank or space character preceding that label. This allows programs like Excel to treat the label as textual information and not a mathematical operation.



Shown on the next page is the directory list for CSV files:



7.5. AERIC Verification Files

There are two types of AERIC verification files generated. These are known as the Format Data files and the Column Data files. These provide the layouts for AERIC for each Table of fixed field files. Format Data files have a file name formed by appending ".FMTDAT.ASCII.TXT" to each rendered fixed field files. Column Data files have a file name formed by appending ".COLDAT.ASCII.TXT" to each rendered fixed field file. All of these files are place in a folder or subdirectory named "AERIC".

The Format Data file contains only one line of text. Shown below is the single line for the Fixed Set table.

```
AMFSA_NIPS_1968_FS___,C,D,1
```

Note that the first field is the file name to be verified with periods changed to "under bar" characters. *(Note: The meaning of the other fields, "C,D,1", are unknown to the NIPSTRAN program as they were specified as required by the AERIC team.)*

The Column Data file contains one line for each column or data field in the file to be verified as identified in the Format Data file above. Below is the Column Data file for the Fixed Set or Table.

```
YEAR,VARCHAR2,Y,0,2,Y,N,N,N
MONTH,VARCHAR2,Y,2,2,Y,N,N,N
CTZ,VARCHAR2,Y,4,1,Y,N,N,N
DIVX,VARCHAR2,Y,5,2,Y,N,N,N
REGT,VARCHAR2,Y,7,2,Y,N,N,N
BN,VARCHAR2,Y,9,3,Y,N,N,N
PCN_,VARCHAR2,Y,12,3,Y,N,N,N
SC0_,VARCHAR2,Y,15,9,Y,N,N,N
VSZ,NUMBER,N,24,4,N,N,N,N
COMSVC,NUMBER,N,28,2,N,N,N,N
COMCMIS,NUMBER,N,30,2,N,N,N,N
COMASGN,NUMBER,N,32,2,N,N,N,N
ADVSVC,NUMBER,N,34,2,N,N,N,N
ADVCMIS,NUMBER,N,36,2,N,N,N,N
ADVASGN,NUMBER,N,38,2,N,N,N,N
ADVPREV,NUMBER,N,40,2,N,N,N,N
PERSASN,NUMBER,N,42,4,N,N,N,N
PERSPFD,NUMBER,N,46,4,N,N,N,N
REPLC,NUMBER,N,50,4,N,N,N,N
RTRNS,NUMBER,N,54,4,N,N,N,N
PERSKIA,NUMBER,N,58,4,N,N,N,N
PERSLST,NUMBER,N,62,4,N,N,N,N
HOSP,NUMBER,N,66,4,N,N,N,N
DESERT,NUMBER,N,70,4,N,N,N,N
AWOLS,NUMBER,N,74,4,N,N,N,N
CAVATT,NUMBER,N,78,4,N,N,N,N
RNGRA,NUMBER,N,82,4,N,N,N,N
RFPFA,NUMBER,N,86,4,N,N,N,N
CIDGA,NUMBER,N,90,4,N,N,N,N
ABNMARA,NUMBER,N,94,4,N,N,N,N
ARTYA,NUMBER,N,98,4,N,N,N,N
COMNAME,VARCHAR2,N,102,10,N,N,N,N
COMGRD,VARCHAR2,N,112,2,N,N,N,N
ADVNAME,VARCHAR2,N,114,10,N,N,N,N
ADVGRD,VARCHAR2,N,124,2,N,N,N,N
HPROV,VARCHAR2,N,126,3,N,N,N,N
HVCMR,VARCHAR2,N,129,2,N,N,N,N
SPROV,VARCHAR2,N,131,3,N,N,N,N
SVCMR,VARCHAR2,N,134,2,N,N,N,N
SEAFID,VARCHAR2,N,136,6,N,N,N,N
UTM,VARCHAR2,N,142,8,N,N,N,N
```

```
STATN,VARCHAR2,N,150,15,N,N,N,N
UNTYP,VARCHAR2,N,165,1,N,N,N,N
QTR,VARCHAR2,N,166,1,N,N,N,N
HY,VARCHAR2,N,167,1,N,N,N,N
```

The first field is the label name followed by the data type used in AERIC verification. The next field indicates if the value must e unique followed by the fields offset and length and whether a value for the field must exist. *(NOTE: The last three "N"s meaning are fixed and unknown to NIPSTRAN.)*

8. References

All the technical manuals used in developing this software product were from the Department of Defense and are available from http://www.dtic.mil/. The Defense Technical Information Center.

(1) Command and Control Technical Center, Computer Systems Manual Number CSM UM 15-78, 1 September 1978, NMCS Information Processing System 360 Formatted File System (NIPS 360 FFS), Users Manual, Volume 1 – Introduction to File Concepts. *Note: Many other NIPS documents can be found here.*

(2) Implementing Hierarchical Data Structures in NIPS, MITRE CORP MCLEAN VA, Cotton,I. W, JUN 1972, Accession Number : AD0763494.

9. Document History

No	Date	Who	What
1	07/09/07	-kdg-	Original Outline begun
2	07/12/07	-kdg-	New Draft Released
3	07/13/07	-kdg-	Version 1 published.
4	07/31/07	-kdg-	Version 2 published. Synchronized with version 2.1.0.0
5	10/10/07	-kdg-	Version 3 published. It Includes column width changes and generate SWL scripts.
6	10/12/07	-kdg-	Added paragraph about column width changes save in "side-car" file.
7	10/19/07	-kdg-	Added details for printing in color and added a tenth column for the data dictionary.
8	10/02/08	-kdg-	Added new table definitions section to data dictionary.

10. Appendices

10.1. Special Use Mnemonics

There are a number of special Mnemonics used by the IBM FFS in the NIPS implementation of that system. These are detailed in the chart below. Note that the rendered output files only render those that have any meaning to the rendered files. The meaning of labels that are not rendered are documented here only for completeness and can be safely ignored if the reader so wishes.

Label	Use	Rendered
+FIL	This element contains the first character in the logical record key which contains the value "R." This character is common to all data records and is used to batch all data records as a block within the OS/360 data set.	No
+RCN	This element contains the total record control group as found in the logical record key. It is not rendered because groups are not rendered. The logical record key to be used in the rendered files is composed of the collection of the Control Set, Subset, or key fields.	No
+PCN	This element contains the set ID field in the key of the logical record.	Yes
+SC*(b)*	I.E. "+SC1", "+SC2", etc. This element redefines the subset control group in the key for a specific subset logical record. For example, for Periodic Set 1, the label is "+SC1+ and its value is "001".	Yes
+BSZ	This element will occur immediately after the key in a logical record and will specify, via binary notation, the number of binary full words within the logical record's binary data block.	No

PSSQ*(n)*	This element definition is generated only for those periodic sets which have not been defined by the user to have a subset control field (based on a data value). It identifies a 4-byte field in the key of a logical record used for subset sequencing within a periodic set. The term (n) represents a one-to-three EBCDIC character suffix used for periodic set identification. I.E. “PSSQ25” will reference the subset sequence field for a logical record of Periodic Set 25.	Yes
VSZ*(n)*	This element is the first binary word in the binary data block of a logical record (fixed set or periodic subset). This binary word will indicate the number of characters currently contained in the logical record’s variable field. If there is no variable field for a logical record, the field will contain zeros.	Yes
VSCTL	This element is a redefinition of VSZ*(n)* above for the logical record containing the first defined variable set.	Yes.

10.2. Error Codes

When NIPSTRAN encounters a situation that it was not prepared for, it will generate an error code in the report file. Those codes are listed below:

Problem Description	Code
Not able to open a needed file. Typically a specified Data Dictionary File.	-1
Program error – trying to interpret a Data Dictionary file that had not been opened.	-2
Data Dictionary file has no data.	-3
There are too many columns on a line in the data dictionary.	-4

There are too few columns on aline in the data dictionary. This could also indicate that there are more than 2047 characters total for this data dictionary entry.	-5
Program error – A logic error has occurred while trying to interpret a line in the data dictionary.	-6
The Mnemonic on the specified line has more than seven characters.	-7
Program error – Table name already set.	-8
Program error – Dataset already set.	-9
Program Error - Mnemonic type already set	-10
Program error – Datatype already set.	-11
Program error – Length already set.	-12
Program error – Code table name already set.	-13
Program error – Name already sey.	-14
Program error – Description already set.	-15
Program error – EC_SOURCE_ALREADY_SET	-16
Program error - EC_PARSE_ERROR_DD	-17
No data data records were found in the Data Dictionary.	-18
Length not set.	-19
Source already set.	-20
Bad tab count.	-21
Can not read line two of section.	-22
Bad data definition section name.	-23
Bad table definition section name.	-24

Unexpected section end.	-25
Not SQL table type.	-26

10.3. NIPS File Structure

The details of NIPS files are too complex and convoluted to detail in this document. The reader is referred to reference (1), Appendix A, Pages 94 – 136.

10.4. IBM EBCDIC Alphameric to ASCII Translation

All EBCDIC characters that are not defined below are not translated. The table below shows the numeric value of an EBCDIC character and the ASCII character rendered by NIPSTRAN. All other values render as NULL. (Note: *Alphameric is IBM's term and not a misspelling.*)

EBCDIC decimal value	Ascii character rendered
64	' '
75	'.'
76	'<'
77	'('
78	'+'
79	'|'
80	'&'
90	'!'
91	'$'
92	'*'
93	')'
94	''
95	Tilda
96	'-'
97	'/'
107	','
108	'%'
109	'_'
110	'>'
111	'?'
122	':'
123	'#'
124	'@'
125	back slash
126	tab
127	'"' quote
129	'a'
130	'b'
131	'c'
132	'd'
133	'e'
134	'f'
135	'g'
136	'h'
137	'i'
145	'j'
146	'k'
147	'l'
148	'm'
149	'n'

150	'o'
151	'p'
152	'q'
153	'r'
162	's'
163	't'
164	'u'
165	'v'
166	'w'
167	'x'
168	'y'
169	'z'
185	'`'
193	'A'
194	'B'
195	'C'
196	'D'
197	'E'
198	'F'
199	'G'
200	'H'
201	'I'
209	'J'
210	'K'
211	'L'
212	'M'
213	'N'
214	'O'
215	'P'
216	'Q'
217	'R'
226	'S'
227	'T'
228	'U'
229	'V'
230	'W'
231	'X'
232	'Y'
233	'Z'
240	'0'
241	'1'
242	'2'
243	'3'
244	'4'
245	'5'
246	'6'
247	'7'
248	'8'
249	'9'

10.5. Zoned Decimal to ASCII Translation

Several standards existed for Zoned Decimal. Each was vendor specific. Since NIPS was based on IBM's Formatted File System, the IBM variant of Zoned Decimal was used for rendering ASCII output. The field was expanded by one character to account for a possible negative sign.

10.6. IBM Mainframe Binary to ASCII Translation

IBM byte order was observed in translating binary values.

10.7. NIPS Military Coordinates to Geographic Latitude-Longitude Coordinates

Although the IBM Formatted File System does not include Geographic Coordinates as a data type, the NIPS system does. The following describes the layout of the NIPS internal data and the output translation. Although the store values are rendered into ASCII as readable text, the internal storage format is documented here as it is found no where else. This format remained a mystery to NARA for decades until a retired military expert revealed the details. The validity of this format was confirmed with several NIPS files that had the values both as coordinate data types and as EBCDIC printable characters.

NIPS also maintained a precision with the field definition. This precision is represented in the rendered ASCII values. The rendered output is as follows:

If precision was 11, then the output is **dddffDDDFFH**.

If precision was 15, then the output could be **dddffffDDDFFFFH**.

Where: **ddd** are degrees from the South Pole

DDD are degrees East or West

FF or **FFFF** is fraction of ONE degree

H has the value **"E"** or "**W**" for hemisphere.

Also if the precision was 15, then the output could be **ddmmsshDDDMMSSH.**

Where: **dd** are degrees north of the equator, only two digits are allotted because the value can not be greater than 90

DDD are the degrees east of the Prime Meriden

mm/MM are the minutes

ss/SS are the seconds

h/H is the hemisphere **N or E**

Stored in the NIPS Coordinate data type are eight bytes (64 bits) of binary data. The layout is shown on the next page. The following is an example of this format with 15 digit precision (**dddffffDDDFFFFH)**:

10669921071778E

Breaking this up into components we have

106 6992 107 1778 E

The 106 value represents 106 degrees above the South pole. If we subtract 90 from that we would have 16 degrees North Latitude. The 6992 is 0.6992 of a degree. To convert into minutes, multiply 60 by 6992 and then divide by 10,000. You get 41.952 minutes. So now we have 106 degrees, 41 minutes and 0.952 of a minute. To convert this to seconds, multiply 60 by 952 and then divide by 1000 yielding 57.12 seconds. So the converted value is 16 degrees, 41 minutes, 57.12 seconds North Latitude.

The same approach is used for Longitude. No subtraction is necessary as the East or West longitude is given.

Below is the detailed chart of the internal storage layout for Coordinates:

Bits	Use or Meaning
1-2 (0-1)	Are not used
3-10 (2-9)	Are the degrees. If this is latitude, the values run from zero (0) which is the South Pole to 180 which is the North Pole. The equator has the value of 90.
11-32 (10-31)	Thousandths of a degree in powers of ½ notation.
33-34 (32-33)	East-West indicator. '01' is East, otherwise West.
35-42 (34-41)	Degrees of longitude.
43-62 (42-63)	Thousandths of a degree in powers of ½ notation.

10.8. Samples of generated SQL scripts for loading rendered data into relational databases

The following shows the SQL generated to load the rendered NIPS file. The database is given the base NIPS file name. Each table is named as intuition would dictate I.E. “FixedTable”, “PeriodicTable1”, etc. System generated labels have leading “+” signs changed to trailing “$” to meet SQL label requirements. The key field is specified as a SQL composite Primary Key. (Note: *It was this Primary Key specification that lead to the discovery of duplicate keys in some NIPS files.) And finally, the location of the generated table file is given. This the user may change when necessary*.)

```
DROP DATABASE IF EXISTS RG349;
CREATE DATABASE IF NOT EXISTS RG349;
 \u RG349 ;

 CREATE TABLE FixedTable (
       USID    CHAR(5) NOT NULL,
       PCN$    CHAR(3) NOT NULL,
       SC0$    CHAR(15)       NOT NULL,
       VSZ     INT     NOT NULL,
       POP     INT     NOT NULL,
       GVNPOP  INT     NOT NULL,
       HOICHN  INT     NOT NULL,
       REF     INT     NOT NULL,
       NVA     INT     NOT NULL,
       VCHAM   INT     NOT NULL,
       VCI     INT     NOT NULL,
       SAS     INT     NOT NULL,
       DAS     INT     NOT NULL,
       PAS     INT     NOT NULL,
       NUMRAD  INT     NOT NULL,
       NUMTV   INT     NOT NULL,
 PRIMARY KEY (
       USID,
       PCN$,
       SC0$
 )

);

/* ------------------------------------------------------------------------------ */
/* NOTE: Change the line below this comment to point to the file on your workstation. */
/*       The file is the tab delimted file for the table identified below.        */
/* ------------------------------------------------------------------------------ */
LOAD DATA INFILE 'C:/NIPS_DATA_WORKING/PSY/TABLES/RG349.PSY.M70TF73.FS___.ASCII.TXT'
INTO TABLE FixedTable
IGNORE 1 LINES;

 CREATE TABLE PeriodicTable1 (
       USID    CHAR(5) NOT NULL,
       PCN$    CHAR(3) NOT NULL,
       DATE1   CHAR(6) NOT NULL,
       CONAG   CHAR(1) NOT NULL,
       REPORT  CHAR(3) NOT NULL,
       OP      CHAR(1) NOT NULL,
       CAMPN   CHAR(1) NOT NULL,
       THEM    CHAR(2) NOT NULL,
       STHEM   CHAR(1) NOT NULL,
       VSZ1    INT     NOT NULL,
       LEAFLET INT     NOT NULL,
       SPRHRS  INT     NOT NULL,
       POSTERS INT     NOT NULL,
```

```
       PUBLS   INT     NOT NULL,
       APT     INT     NOT NULL,
       CAP     INT     NOT NULL,
       CDT     INT     NOT NULL,
       CVC     INT     NOT NULL,
       HB      INT     NOT NULL,
       HE      INT     NOT NULL,
       MDC     INT     NOT NULL,
       RDC     INT     NOT NULL,
       SLT     INT     NOT NULL,
       VIS     INT     NOT NULL,
       AUD     CHAR(1) NOT NULL,
 PRIMARY KEY (
       USID,
       PCN$,
       DATE1,
       CONAG,
       REPORT,
       OP,
       CAMPN,
       THEM,
       STHEM
 )

);

/* ---------------------------------------------------------------------------- */
/* NOTE: Change the line below this comment to point to the file on your workstation. */
/*       The file is the tab delimted file for the table identified below.      */
/* ---------------------------------------------------------------------------- */
LOAD DATA INFILE 'C:/NIPS_DATA_WORKING/PSY/TABLES/RG349.PSY.M70TF73.PS001.ASCII.TXT'
INTO TABLE PeriodicTable1
IGNORE 1 LINES;

 CREATE TABLE PeriodicTable2 (
       USID    CHAR(5) NOT NULL,
       PCN$    CHAR(3) NOT NULL,
       DATE2   CHAR(4) NOT NULL,
       REPUNIT CHAR(3) NOT NULL,
       VSZ2    INT     NOT NULL,
       RNEWS   INT     NOT NULL,
       RENT    INT     NOT NULL,
       RIP     INT     NOT NULL,
       TVNEWS  INT     NOT NULL,
       TVENT   INT     NOT NULL,
       TVIP    INT     NOT NULL,
 PRIMARY KEY (
       USID,
       PCN$,
       DATE2,
       REPUNIT
 )

);

/* ---------------------------------------------------------------------------- */
/* NOTE: Change the line below this comment to point to the file on your workstation. */
/*       The file is the tab delimted file for the table identified below.      */
/* ---------------------------------------------------------------------------- */
LOAD DATA INFILE 'C:/NIPS_DATA_WORKING/PSY/TABLES/RG349.PSY.M70TF73.PS002.ASCII.TXT'
INTO TABLE PeriodicTable2
IGNORE 1 LINES;
```

10.9. Comments about the source files and their location

This was not a managed project. To “discover” if the metadata was in the NIPS files, a simple “C” program for DOS was quickly written. This grew into a discovery process as more and more of the metadata was “exhumed.” At the point where it was obvious that a solution could be crafted, the code was then wrapped into a Windows C++ project and extended as each issue was addressed. Hence there is a lack of design documents, or real design.

The source code, executable, and Microsoft Project files are located on the shared drive “S:” (Novell Ns on ‘A2share2\Data\Shared’) under S:\NWME\NIPSTRAN.

[-End-]