For Presentation to the Fault & Disturbance Analysis Conference at Georgia Tech Atlanta, Georgia; April 26th, 2005

File Naming Convention for Time Sequenced Data (TSD) - Part II

Sponsored by the Power System Relaying Committee (PSRC) of the IEEE Power Engineering Society (PES)

H8 Work Group: Mark Adamiak, Scott Anderson, John Chadwick, Rick Cornelison, Ratan Das, Anthony Eshpeter, Tony Giuliante, Erich Gunther, Jim Hackett, Jim Ingelson, Robert Johnson, Mladen Kezunovic, Amir Makki, Maria Makki, Ken Martin, Pierre Martin, Harish Mehta, Mike Meisinger, George Moskos, Krish Narendra, Robert Orndorff, Jeff Pond, Mohindar Sachdev, George Semati, Larry Smith, John Sperr, Bill Strang, Mark Taylor, Don Wardlow, Miguel Xavier

Abstract: A procedure for naming time sequence data (TSD) files such as transient data records, event sequences, and periodic data logs is recommended. Sources of TSD files are described and a survey of current naming techniques is provided. The advantages of using a common naming procedure are highlighted and the limitations and applications are identified. Issues of compatibility across operating systems and various vintages, and adaptability to other types of files are also discussed. The required and optional portions of the naming procedure are described in detail and many examples are provided.

Keywords: time sequence data, TSD, filename, file name, file extension, file naming convention, comma separated values, CSV, required fields, user fields, start date, start time, fault data, fault time, time code, station identifier, device identifier, company name

Definition: A TSD file is any type of electronic data file where each data item in the file corresponds to an instant of time that is identified by an explicit or implicit time tag, such as, transient data records, event sequences, and periodic data logs.

1. Introduction

Microprocessor based measurement and protection devices (digital devices) used in electric power substations produce large quantities of TSD files that contain vast volumes of data about the power system. Additional TSD files are also produced while processing the original data, and while performing maintenance and testing operations on the original devices. There are many problems associated with handling such large quantities of TSD files. The term handling means reporting, saving, archiving, exchanging and so on. The problems are mainly due to the fact that currently there are too many different types of proprietary naming conventions in circulation.

Work on this project began with the IEEE-PES-PSRC Communications Subcommittee task force HTF8 in 1999. Working group H8 then followed and their report was posted in 2001. Their reported naming convention solved many of the problems that are associated

with handling the large quantities of TSD files. Their main objective was to define a common naming convention. In 2002 a new task force was formed to study the potential of developing the common naming convention into an IEEE recommended practice. In 2003 the above working group was formed to write the recommended practice.

The recommended filename is human readable and includes, among other features, key portions of the information contained in the file including, but not limited to, the name of the circuit, substation and recording device, and the date and time of initial occurrence. This was not easy to do under DOS since the limit was 11 characters per filename, but now filenames of up to 253 characters are generally permitted.

The recommended naming convention has been gaining popularity and is being used by a nontrivial number of utilities, independent system operators, and manufactures. The North American Electric Reliability Council (NERC) and the Northeast Power Coordinating Council (NPCC) have also recommended the use of a common naming convention.

1.1 Overview

Filenames are essential for both operating system and user. The filename is the system's key for unlocking the contents, and without friendly names (where key information about the file is in the filename) the user will have trouble handling large numbers of files. Also, programs for analysis and trending applications have to automatically sift through and process large numbers of files. Reading file contents requires considerable disk access time especially for large files. But, reading filenames is much faster because the filenames are stored separately from their contents in system files called allocation tables and these tables can be quickly loaded with minimal disk access time.

Meaningful filenames provide software developers with the ability to write programs that can quickly manage and process large numbers of files. The hardware benefits too from the reduced number of disk access operations. The alternative to meaningful filenames is to build and maintain a specialized database. However, these specialized databases are very costly to create, require substantial programming support, produce compatibility issues, and use extremely large memory structures.

Accordingly, the filename information is specified in comma-delimited format where commas are used to separate the information into multiple fields. Thus, spreadsheet like tables can be made from directory listings of filenames. These tables provide users with an easy way to perform sort and query operations based on any one of the fields in the filename, in effect, providing the same look and feel as any other real life database.

1.2 Purpose

The purpose of this recommended practice is to define a procedure for naming TSD files; a procedure that is needed to resolve many of the problems that are associated with

reporting, saving, exchanging, archiving and retrieving large numbers of files. There is no other defined standard for naming such files at this time.

2. Filenames

A filename is composed of two parts. The first part is the name and the second part is the extension. The name and extension parts are separated using the dot “.” symbol. The extension is normally used to specify the file type. There are two types of files: the first type is binary, such as, program files (.EXE) and dynamic link libraries (.DLL), and the second type is ASCII, such as, initialization files (.INI), comma separated values (.CSV), and text files (.TXT).

Filenames are listed in hidden files called allocation tables (file allocation table "FAT" for DOS, and NT File System "NTFS" for Windows NT). Each entry in the allocation table corresponds to a single file and has a number of fields including the filename, the attributes ("A" for archive and "H" for hidden), and the beginning memory address where the file’s contents are stored. So far, the DOS style filenames have been the most popular and the most restrictive. The 95 and NT generation of Windows significantly relaxed the DOS restrictions. A Windows filename can be up to 253 characters long (11 for DOS) and each filename character can be any one of 245 ASCII codes (52 for DOS).

2.1 Naming Conventions

A number of naming conventions are in use today. These formats can be organized in three classes. The classes are associated, coded, and sequenced.

Associated means that the filename extension defines the type of data storage format. For example, the extensions "HDR", "CFG", "DAT", and "INF" are used to indicate that the file contents are compatible with the IEEE C37.111 COMTRADE standard. The non-extension part of an associated filename is totally up to the user.

Coded means that the filename contains some information about the event. In this case, the storage format is usually manufacturer specific. For example, certain files that are generated from digital fault recorders have the event date and time (up to 12/31/2079-23:59:59.99) and the recorder number (up to 255) coded in the filename. The recorder number is coded in the first 2 characters of the name and the date and time are coded in the last 9 characters of the filename. The resulting filename is not friendly and reading it requires special decoding software. For example, “G30BQ1EF.063” is the filename assigned by device number 163 on 09/18/1991 at 14:15:00.630. A detailed example of a similar scheme used to compress over 70 characters of key event information into the DOS 8.3 filename format is presented in Appendix-A.

The sequenced filenames format is an incremental approach to naming files. This method is valid because the resulting filenames are unique. The sequence may appear in the

name or in the extension portion of the filename. The total number of attainable filenames is limited to the maximum value of the numerical sequence. When multiple devices are used then the device numbers are also coded in the filename. For example, some filenames have the location name (up to 4 characters), the event number (up to 4 characters, 9999 filenames before overwrite) and the channel group number (up to 3 characters). A file named "MART1743.RCL" indicates that the event was recorded at the Martin Station and that the event number is 1743. The "RCL" extension means that the file contents are from the first 16 analog input channels.

2.2 Coding Schemes

Filenames are limited in length. Compression, especially in DOS where only 11 characters are allowed, may be needed in order to place required key information in the filename. Compression is the art of representing a long sequence of information with a brief sequence of codes. Popular data compression schemes include but are not limited to run-length coding, half-byte packing, Huffman coding, dictionary coding and adaptive dictionary coding.

Run-length coding is a data compression technique in which repeated symbols are counted and then replaced by the resulting count. For example, the sequence CDEFFFFGGGGGGG333333333 will reduce to CDE^F4^G7^39. The technique does not always work. For example, ABCDEFGH-XYZ-0123456789 will not compress.

Half-byte packing is a procedure used to mask out repeated patterns in character bits. For example, the characters 7 and 14 represented by the bit patterns 0000-0111 and 0000-1110 can be packed in one byte by masking out the upper 4 bits from both characters. The resulting byte will have the bit pattern 0111-1110. Character bits are not always repetitive and bit manipulation is not natural for byte-based machines.

Huffman coding is a simple method where the probability or relative frequency of the occurrence of each symbol in a stream of information is used to compress the information. Symbols with highest probability are assigned to codes with shortest length. For example, if the term "filename" appears frequently, then it can be assigned the code "☺" which saves 7 characters per appearance.

Dictionary and adaptive dictionary coding are compression methods that are similar to Huffman coding. Instead of assigning short codes to frequent terms or phrases, terms are replaced by pointers to their location in a dictionary or a list of unique entries. Repeated terms are reduced to repeated pointers. In the adaptive method, repeated terms are replaced with pointers to their initial place of occurrence. For example, PKZIP, a popular file compression program, uses a similar adaptive technique.

The above methods are designed to compress information using the standard ASCII character set. However, a number of ASCII characters (such as " ? / \ < > * | :) are not allowed in the filename. Accordingly, applying these compression methods could produce

non-valid filenames. To eliminate this problem an alternate code set is used. The set is composed of the ASCII characters that are allowed in the filename. The alternate code set is normally called the “filename character set”.

3. Recommended File Naming Convention

The file naming convention defines a readable, delimited filename format. The delimiting character between the filename fields is the “,” comma. In all cases where an alphabetical character is called for, the character can be either upper or lower case. Software should treat upper and lower case letters in the same way. The fields for the filename shall be as follows and in order as shown here:

Start Date,Start Time,Time Code,Station Identifier,Device Identifier,Company Name

The above fields are called “required fields”. Additional fields may be added as needed by the user and are called “user fields”. The convention requires that the user fields follow directly after the required fields in order as shown here:

,User 1,User 2,User 3,and so on.Extension

All required and user fields should be separated by commas. Only one comma should be used to separate between two fields. Trailing commas should not be used. The filename extension will always follow at the end as shown above. In order to conform to this convention the required fields must appear in the filename in order as specified above. The user fields may be used and if so they should be used in order as specified above. The extension field (including the “.” dot) may be preceded with a comma as shown below for compatibility purposes with spreadsheet type programs:

,User 1,User 2,User 3,and so on,.Extension

3.1 Limitations

In addition to the previously mentioned limitations regarding the length and the allowable characters of the filename, additional limitations exist while trying to copy long filenames onto CD and floppy disks. Specifically: floppies are limited to a very small number of long filenames per disk regardless of file size; current CDs do not work with filenames that are more than 64 characters long; users who deal with thousands of long filenames per folder have also reported problems with their zipping applications. However, given the rate of advancement in technology it is highly possible that these limitations will soon vanish or become obsolete. The naming convention has been successfully tested under a variety of operating systems including: Windows, Linux, Unix, and Novell.

3.2 The Required Fields

Start Date: the field is defined in a numeric format that allows for sorting. The field width is six (6) digits: the first two (2) digits are for the year, the second two (2) digits are for the month, and the last two (2) digits are for the day. For example, the code 010203 means the date is February 3rd, 2001. The Start Date field is the date at which the first sample was recorded. Depending on the last character in the Time Code field, the first sample could mean physically the first sample in the file, or it could mean the first sample in the fault record portion of the file (the trigger point). This distinction is made because digital relays and digital fault recorders insert a pre-fault record before the actual fault record. It is up to the user to decide whether the date of the first sample comes from the file, or comes from the fault record. By default, the Start Date is the date of the first sample in the file. If the last character of the Time Code field is set to “t”, then the Start Date is the date of the first sample in the fault record.

Start Time: the field is defined in military time format and can be specified to the required precision. For example, 170215183222, 170215183, 17021518, 170215, and 1702 are all acceptable times. The Start Time field is the time of day of the first sample. Depending on the last character in the Time Code field, the first sample could mean physically the first sample in the file, or it could mean the first sample in the fault record. By default, the Start Time is the time of the first sample in the file. If the last character of the Time Code field is set to “t”, then the Start Time is the time of the first sample in the fault record.

Time Code: the field is restricted to a maximum of seven (7) formatted characters. The first character is the sign and is followed by up to five (5) characters indicating the time difference between the time system used for the Start Date and Start Time (the filename’s time tag) and Universal Time (UT, also called Greenwich Mean Time or “Zulu” time). The format is up to two (2) digits for the hours, followed by an optional letter “h” with up to two (2) digits for the minutes. For example, the code +10h30 means the time difference is 10 hours and 30 minutes (half hour time zone), and the plus sign means that time tag is ahead of UT. If UT is used for the time tag, this field will contain the two letters “UT”. The last character in this field is also an optional letter “t” and is used to indicate that the time tag is not referencing the first sample in the file but is actually referencing the trigger time in the fault record. Examples are as follows:

-4	*time tag is 4 hours behind UT*
+5t	*time tag is 5 hours ahead of UT and references trigger time*
-7h15	*time tag is 7 hours and a quarter behind UT*
+10h30t	*time tag is 10 hours and a half ahead of UT and references trigger time*
UT	*time tag is UT*
UTt	*time tag is UT and references trigger time*

In addition, the calculation for the time difference should also consider whether standard time or daylight time was in affect at the time of the recording. The result after this consideration should be the one reported in the filename. However, this consideration is not required if the originating digital instrument automatically adjusts to standard or

daylight at the conventional times of the area.

Station Identifier: users can formulate their own code for the station where the originating device is located (a unique name within the company). This is a variable length field and can contain letters, numbers, and some punctuation marks. Characters disallowed are , ? “ / \ < > * | : (i.e. comma, question mark, quotation mark, forward slash, backward slash, less than, greater than, asterisk, pipe, and colon).

Device Identifier: users can formulate their own code for the originating device (a unique name within the station). This is a variable length field and can contain letters, numbers, and some punctuation marks. Characters disallowed are , ? “ / \ < > * | : (i.e. comma, question mark, quotation mark, forward slash, backward slash, less than, greater than, asterisk, pipe, and colon).

Company Name: users can formulate their own code for the company that operates the originating device (a unique name within the corporate realm). This is a variable length field and can contain letters, numbers, and some punctuation marks. Characters disallowed are , ? “ / \ < > * | : (i.e. comma, question mark, quotation mark, forward slash, backward slash, less than, greater than, asterisk, pipe, and colon).

3.3 The User Fields

One (1) or more optional fields are allowed, as needed, as long as there is room available in the filename. These fields are called user fields and can be used for comments or for any other purpose that is desired. The user fields are variable length fields and can contain letters, numbers, and some punctuation marks. Characters disallowed are , ? “ / \ < > * | : (i.e. comma, question mark, quotation mark, forward slash, backward slash, less than, greater than, asterisk, pipe, and colon).

Having these user-defined fields should not be a problem to the originators of the file, because they obviously know which field(s) they are using. Upon passing the file to another party, that other party would not be able to automatically identify the user field(s), but they could display them and could likely recognize them by a quick inspection of their contents. The philosophy behind this is that the most significant items have been made required fields. Another factor is trying to limit in every possible way the number of characters used in the filename.

The following user fields are optionally recommended for packing additional meaningful information in the filename:

Duration: the duration is equal to the time difference between the first and last samples in the file. This time difference can be for up to one millisecond or up to a number of years. Two (2) user fields may be required in order to adequately support the duration value. The first field should be formatted in the same way as the Start Date field, and the second field should be formatted in the same way as the Start Time field. The duration value can

therefore be specified to the precision necessary, leading zeros are necessary. For example, a file with 99 seconds of data will have a duration code of “000000, 000139”. A file with 2 years, 1 day, and 99 seconds will have a duration code of “020001, 000139”.

Type: this field may be used to describe the type of originating event. For example, the code “AG” can be used to indicate that the file contains data for a Phase-A to Ground fault. Or: the code “LOG” can be used to indicate that the file contains periodic readings of load and other data (such as temperature); the code “SER” can be used to identify a sequence of events file; the code “TST” can be used to specify that the file was created for relay testing and maintenance operations.

Geographic Position Coordinates: two (2) user fields are required in order to support position information. The first field is an expression of latitude in degrees, a comma delineator, and the second field is an expression of longitude in degrees. Leading zeros are necessary. These fields can be carried to the resolution desired as per the discussions on position information shown below in section 3.6.

3.4 The Extension Field

Extension: the exact field definition is up to the user. However, the extensions: .DAT, .CFG, .HDR, and .INF are reserved and used to indicate that the file is compatible with the IEEE C37.111 COMTRADE standard. There are no other reserved extensions, however, the previously noted restrictions on filename extensions should, at all times, be observed. For example, users should avoid using .EXE, .BAT, .SYS, .COM, .DLL, or any other commonly known extension for naming TSD files.

3.5 Examples of the File Naming Convention

Beside each of the example filenames that follow below we have placed the total number of characters, without including the dot character in the count. The following is an example filename using only the required fields. The Start Time field is given to a millisecond resolution. It is a COMTRADE configuration file:

000809,175215183,-4,sta80,ben717,nyiso.cfg (41)

The following is another example but with the Start Time field given to a tenth of a second resolution and with three (3) additional user fields for duration (95.9 seconds) and type:

000809,1752152,-4,sta80,ben717,nyiso,000000,0001359,uf.cfg (57)

The following example filename includes an additional user field for comments:

000809,1752152,-4,sta80,ben717,nyiso,000000,0001359,uf,critical-frequency.cfg (76)

3.6 Geographic Position (Latitude & Longitude)

Using the location coordinates will absolutely assure uniqueness of the filename. Latitude and longitude is the most commonly used position coordinate system. Latitude is always expressed first. The letter “n” indicates latitude in the Northern Hemisphere. The letter “w” indicates a longitude in the Western Hemisphere. These letters can appear on either side of the expression. Latitude and longitude are expressed in decimal degrees. For degrees less than 100, leading zeros should be included.

Each degree of latitude covers 69 (statute) miles. Each degree of longitude covers 69 miles at the equator and less as you move toward the poles. Therefore .001 degrees covers 364 feet for latitude or 364 feet or less for longitude, which is more than adequate resolution for a station location. GPS position fixes with civilian equipment became more accurate and repeatable in May of 2000 when selective availability (SA) was turned off. SA was an intentional dithering of the GPS signal. Using a hand held civilian GPS receiver (after SA was turned off) at opposite corners of a large transmission substation gave these results: Entry Corner = 042610n, 073905w; and Opposite Corner = 042608n, 073903w.

5 GPS position fixes were taken at a fixed location over a period of several weeks after SA was turned off, and it was observed that the repeatability was better than .001 degree. These results suggest that it is reasonable to specify station location using either 2 or 3 decimal places but not 4. However, applications should accommodate whatever number of decimal places used. The expression 04261n, 07390w is acceptable for the above station position. The suggestion of using location coordinates may be a problem for many utility users because they have security concerns of releasing sensitive information such as the location of their substations. If a utility would like to include location coordinates then they may consider “encoding” the information and placing it in the user fields.

3.7 Additional Examples

Here is the same example as above but with position information in the suggested form:

000809,175215183,-4,sta80,ben717,nyiso,04305n,07767w.cfg (55)

The following is another example with more users fields included:

000809,175215,-4,sta80,ben717,nyiso,04305n,07767w,log,summer-overload.cfg (72)

3.8 Short Filenames

Naming a file is very much like naming a chapter in a book or a section in a paper. The intent is to define a short term or phrase that describes the contents. It is therefore recommended for users of this convention to always strive to create filenames that are

short and sweet. To assist in the manual assignment of TSD filenames and to allow for their use for other purposes (other than naming data originating from a specific device) abbreviated fields are highly recommended, and the use of too many user fields should be avoided. The following is an example of a short filename for a user generated file representing calculated swings in system frequency:

030914,160404,ut,sys,hz,sce.xls (30)

Notice that in this case the Station Identifier field is set to “sys” (short for power system), and the Device Identifier field is set to “Hz” (short for hertz). This example filename is acceptable and is in full compliance with the convention. The use of short names is very important especially for readability and exchange across various types of storage media, communication links, and operating systems. Accordingly, the recommended practice strongly recommends that users of this convention aspire to creating filenames that are always less than 64 characters long.

4. Uniqueness

Uniqueness is essential in electronic filing systems. Uniqueness depends on the choices made by the user. The names are absolutely guaranteed to be unique if the geographic position information is included. However, users of this convention should assign their filenames such that the required fields alone can provide a sufficient guarantee that the filenames are unique.

5. Universality

The intent of the recommended practice is to address the issue of naming TSD files. The fact that the recommended practice may also be used for naming other types of files is an unintentional byproduct of the original intent, such practices are strongly recommended.

The nature of the format is self-readable, that by just looking at the user fields one can decipher their actual types. Nevertheless, the extensible markup language (XML) format is recommended for defining user fields if they exist. The XML file should be named “COMNAMES.DEF” and should list the required fields on top as a standard template format. The user fields can then be defined under the required fields in their order of appearance in the filename. An example of the recommended XML definition file is shown below. The definition file includes an object definition for each one of the six (6) required fields, plus one (1) object definition for the recommended user field “Type”:

C:\TSD>TYPE ***COMNAMES.DEF***
<file definition>
    <definition name>TSD Files</definition name>
    <file pattern>.cfg;*.dat;*.inf;*.osc</file pattern>*
    <number of fields>7</number of fields>

<field1 name>Start Date</field1 name>
<field1 type>date</field1 type>
<field1 format>yymmdd</field1 format>
<field2 name>Start Time</field2 name>
<field2 type>time</field2 type>
<field2 format>hhMMssmmm</field2 format>
<field3 name>Time Code</field3 name>
*<field3 type>string*7</field3 type>*
<field4 name>Station Identifier</field4 name>
*<field4 type>string*6</field4 type>*
<field5 name>Device Identifier</field5 name>
*<field5 type>string*24</field5 type>*
<field6 name>Company Name</field6 name>
*<field6 type>string*12</field6 type>*
<field7 name>Type</field7 name>
*<field7 type>string*3</field7 type>*
</file definition>

Notwithstanding the above, users of this convention should not mishmash multiple types of naming conventions within the same folder. Any future conventions should adhere to this CSV format. In this way, subsequent conventions can define the appropriate types of fields for the appropriate types of applications. Users wanting to define their own filename formats would simply develop the definition in the specified format as shown above. The procedure defined herein is therefore universally applicable to all file types.

6. Applications

There are many applications that can be realized given a unique, informative file naming system. Possible suggested applications are: time line manager - TSD file manager, universal TSD viewer, TSD analysis system, power system stability monitor, global-international data bank for TSD files, and so on.

Among the main beneficiaries of the proposed convention are the utilities and the emerging global and local electric reliability commissions. Utilities, independent operators, manufactures, and reliability commissions can now realize the benefits of sharing a common electronic filling system. Take any TSD file generated from any device by any company anywhere on the planet and seamlessly file it. A recommended file naming convention by IEEE is therefore necessary and is accordingly provided in this document.

Appendix–A, Example Coding Algorithm

The key event information, or the keys that will be coded in the filename are severity measure (from 0 to 9), date and time (from 1/1/1980-00:00:00.000 to 12/31/2079-23:59:59.999), substation name (up to 24 characters), recorder type and manufacturer (up

to 24 characters), and recorder number (from 0 to 9999).

The worst-case scenario is to compress all of the above keys, up to 74 characters total, down to 11 filename characters in DOS. In that case, the "/", "-", ":", and "." characters of the date and time keys are known symbols and can be ignored. Thus, only 68 characters have to be compressed down to 11 DOS filename characters.

The substation name, the recorder type, and the recorder number keys account for 52 of the remaining 68 characters. Their values will remain constant for all of the files that are generated by the same recorder. In other words, they are repeated terms. They can be combined and replaced by a pointer to an entry in a dictionary or in a list of recorders. Thus, 52 of the 68 characters are replaced with 1 pointer. Thus, only 16 characters and 1 pointer have to be compressed down to the 11 filename characters of DOS.

A DOS filename character can be any one of 52 available codes. Clearly, the month (1 to 12), day (1 to 31), and hour (1 to 23) keys have values that are under 52 and can be represented with 3 separate filename characters, which means, 6 of the 16 characters compress down to 3 filename characters. Thus, 10 characters and 1 pointer are left and must fit in 8 filename characters.

The minute (up to 59) and second (up to 59) can not be compressed with 2 separate filename characters. These keys can reach values that are above the available 52 codes per character. However, by utilizing the first 32 codes to emulate a 5-bit register, any filename character can be used as an array of up to 5 flags. A value of up to 104 can be compressed down to 1 filename character and 1 flag. If the value of the key exceeds 52, then the value minus 52 is coded and the flag is set. If the value does not exceed 52, then the value is coded and the flag is reset. The 4 characters of the minute and second keys compress down to 2 filename characters and 2 flags. Thus, 6 characters and 1 pointer are left to compress and room for only 5 more filename characters and 3 flags is left.

The severity measure (up to 9), year (up to 99 years from the year 1980), and millisecond (up to 999) keys compose the final 6 characters. These characters can be aligned to form 3 numbers, each being 2 digits wide, which will compress down to 3 filename characters and 3 flags. Thus, only 1 pointer remains and there is room left for 2 more filename characters.

Unfortunately, the pointer must be able to index up to 9999 dictionary entries, which will require the 2 remaining filename characters plus an additional 2 flags. Just 2 more flags are needed in order to compress the 74 ASCII characters down to the 11 DOS filename character format.

One more trick is required to procure the 2 extra flags: Since the month and hour keys have values that are always under 26 and since the 52 codes can be divide into 2 groups of 26 codes each, then the month and hour keys can be assigned to codes that are either from the lower or from the upper group. And that takes care of the 2 extra flags.

In conclusion, the 11 characters of a DOS filename do provide the space needed to compress up to 74 characters of key event information. However, this type of compression does not normally result in friendly filenames. The frequent user will need a program to decode and display the key information. Source code to automate the above algorithm is listed in Appendix-B.

Appendix–B, Example Source Code

The following source code listing is taken from a Pascal program that encodes and decodes up to 74 characters of key event information to and from the 8.3 DOS naming convention.

The authors of this program are members of the IEEE-PES-PSRC File Naming Convention Working Group and have elected, at their own discretion, to place this code in this recommended practice document. Users of this convention may use copies of the source code listed herein or portions thereof with their own applications provided that this original notice is not removed from any of the copies made, taken, or translated.

```
const
  code_str:
    string[51]='0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ_^$~!#%&-{}()@`';

type
  wrk12:string[12];
  dfdname_info=record
    rid,id,jday,year,day,month,hr,min,sec,msec,ftyp:integer;
  end;

var
  dfdname:dfdname_info;

function char_str(intval:integer):char;
begin
  if intval in [0..49] then
  begin
    char_str:=code_str[intval+1] else char_str:='`';
  end;
end;

function char_val(intstr:char):integer;
var viv:integer;
begin
  viv:=1; while ((intstr<>code_str[viv]) and (viv<50)) do inc(viv);
  if intstr=code_str[viv] then char_val:=viv-1 else char_val:=51-1;
end;
```

```
function encode_basechr:wrk12;
var p,y,q,m,s,d,t,shiftchr:integer; dfdfnam:string[12]; upflag:byte;
begin
  shiftchr:=50; dfdfnam:='````````.```';
  with dfdname do
  begin
    if ((rid>=129) and (rid<=132)) then rid:=rid-32;
    if rid>=shiftchr then p:=1 else p:=0;
    if (id div 100)>=shiftchr then y:=1 else y:=0;
    if (id-100*(id div 100))>=shiftchr then q:=1 else q:=0;
    if min>=shiftchr then m:=1 else m:=0;
    if sec>=shiftchr then s:=1 else s:=0;
    if ftyp<5 then
    begin
      d:=0; t:=ftyp;
    end else
    begin
      d:=1; t:=ftyp-5;
    end;
    dfdfnam[1]:=char_str(rid-(p*shiftchr));
    dfdfnam[2]:=char_str((id div 100)-(y*shiftchr));
    dfdfnam[3]:=char_str((id-100*(id div 100))-(q*shiftchr));
    dfdfnam[5]:=char_str(year-1980);
    dfdfnam[6]:=char_str(month+36);
    dfdfnam[7]:=char_str(day);
    dfdfnam[8]:=char_str(hr+(q*(shiftchr div 2)));
    dfdfnam[10]:=char_str(min-(m*shiftchr));
    dfdfnam[11]:=char_str(sec-(s*shiftchr));
    dfdfnam[12]:=char_str(((msec div 10)+(t*10)));
    if p<>0 then upflag:=$10 else upflag:=$00;
    if y<>0 then upflag:=upflag or $08;
    if m<>0 then upflag:=upflag or $04;
    if s<>0 then upflag:=upflag or $02;
    if d<>0 then upflag:=upflag or $01;
    dfdfnam[4]:=char_str(upflag);
  end;
  encode_basechr:=dfdfnam;
end;

procedure decode_basechr(dfdfnam:wrk12);
var p,y,q,d,m,s,t,shiftchr,upflag:integer;
begin
  shiftchr:=50;
  with dfdname do
  begin
```

```
    upflag:=char_val(dfdfnam[4]);
    if ((upflag and $10)=$10) then p:=1 else p:=0;
    if ((upflag and $08)=$08) then y:=1 else y:=0;
    if ((upflag and $04)=$04) then m:=1 else m:=0;
    if ((upflag and $02)=$02) then s:=1 else s:=0;
    if ((upflag and $01)=$01) then d:=1 else d:=0;
    if char_val(dfdfnam[8])>=(shiftchr div 2) then q:=1 else q:=0;
    rid:=char_val(dfdfnam[1])+(p*shiftchr);
    if ((rid>=97) and (rid<=100)) then rid:=rid+32;
    id:=(100*(char_val(dfdfnam[2])+(y*shiftchr)))+(char_val(dfdfnam[3])+(q*shiftchr));
    year:=char_val(dfdfnam[5])+1980;
    month:=char_val(dfdfnam[6])-36;
    day:=char_val(dfdfnam[7]);
    hr:=char_val(dfdfnam[8])-(q*(shiftchr div 2));
    min:=char_val(dfdfnam[10])+(m*shiftchr);
    sec:=char_val(dfdfnam[11])+(s*shiftchr);
    t:=char_val(dfdfnam[12]) div 10;
    msec:=10*(char_val(dfdfnam[12])-(t*10));
    ftyp:=t+(d*5);
  end;
end;
```

Appendix–C, Bibliography

This listing of books, articles & standards is provided as sources for additional information:

“Washington View: SA - Going the Way of the Dinosaur,” Davis, D. A., GPS World, V. 11, No. 6, June 2000, pages 16-20

"IEEE Standard Common Format for Transient Data Exchange (COMTRADE) for Power Systems," IEEE Standard P37.111, 1991, 1999

"Survey of File Naming Schemes,” A. Makki, Fault and Disturbance Conference, Georgia Tech, May 1998

“GPS Land Navigation,” Ferguson, Glassford Publishing, Boise, Idaho, 1997

"Overview of FAT, HPFS, and NTFS File Systems,” Win NT Resource Kit, Chp18, 1997

"Data Compression in Digital Systems," Huffman, Chapman & Hall, New York, 1997

"The Data Compression Book," Nelson, M&T Publishing, California, 1992

"Working with MS-DOS files," Microsoft User Guide & Reference MS-DOS 5.0, 1991