Submitted to the 9th Annual Fault and Disturbance Conference at Georgia Tech
May 1st, 2006; Atlanta, GA

COMBINING DIGITAL FAULT RECORDS FROM VARIOUS TYPES OF DEVICES (VIRTUAL-DFR)

Draft 1.0, By:
Amir Makki, Maria Rothweiler Makki (SoftStuf, Inc)
Jeff Pond (National Grid)
Alexander Apostolov (Areva, T&D)

I - SUMMARY

The art of combining digital fault records together to form a single record has been around for many years. The resulting single record is usually called the composite or virtual record and is always center around the fault area containing only the common time data. The resulting combined record is normally saved in the standard IEEE C37.111 Comtrade format.

The subject is of renewed interest because of the recent need for combining records from digital relays and presenting them as though they were extracted from a single digital fault recorder (DFR), hence the term “Virtual DFR”. The process of combining fault records is very complex because there are too many types of measurement and protection devices being manufactured and used. These devices produce different types of fault records and their sampling methodologies and data formats vary depending on the manufacturer and on the type of device.

The paper provides a number of examples derived from actual real life events. The paper starts with a short survey of the various types of digital fault records, and then describes an automatic process for combining such records. The paper also reveals the key issues that are essential for assuring quality and speed. Special emphasis is on channel definition databases and analysis procedures that are essential for combining various types of digital fault records, including interpretation of various types of data formats, matching frequencies (interpolation, extrapolation and re-sampling) and aligning data samples in time, scaling factors, and calculating missing phases. The intent of the authors is for the paper to serve as a guide for combining fault records from multiple devices.

II - BACKGROUND

Combining digital fault records is about organizing data samples in a related, logical way that simplifies the task of trying to determine what happened and why. Ever since the early days of digital fault recording developers have been building applications to automate or simplify the process of combining records. Historically, the process has

been an art form. The design methodologies vary depending on the type of application. As for this paper, the main question is: how do we combine fault records from various types of devices? Before we address this question let us briefly examine the nature of fault records.

Formats: Fault records originate from various types of digital devices, such as relays and fault and disturbance recorders. Fault records are saved in computer files and their formats are classified as either proprietary or standard.

Proprietary: There are too many types of proprietary data formats in circulation today (ASCII and Binary). The formats vary depending on the manufacturer and on the type of originating device. Figure-1 shows a few examples of proprietary record formats.

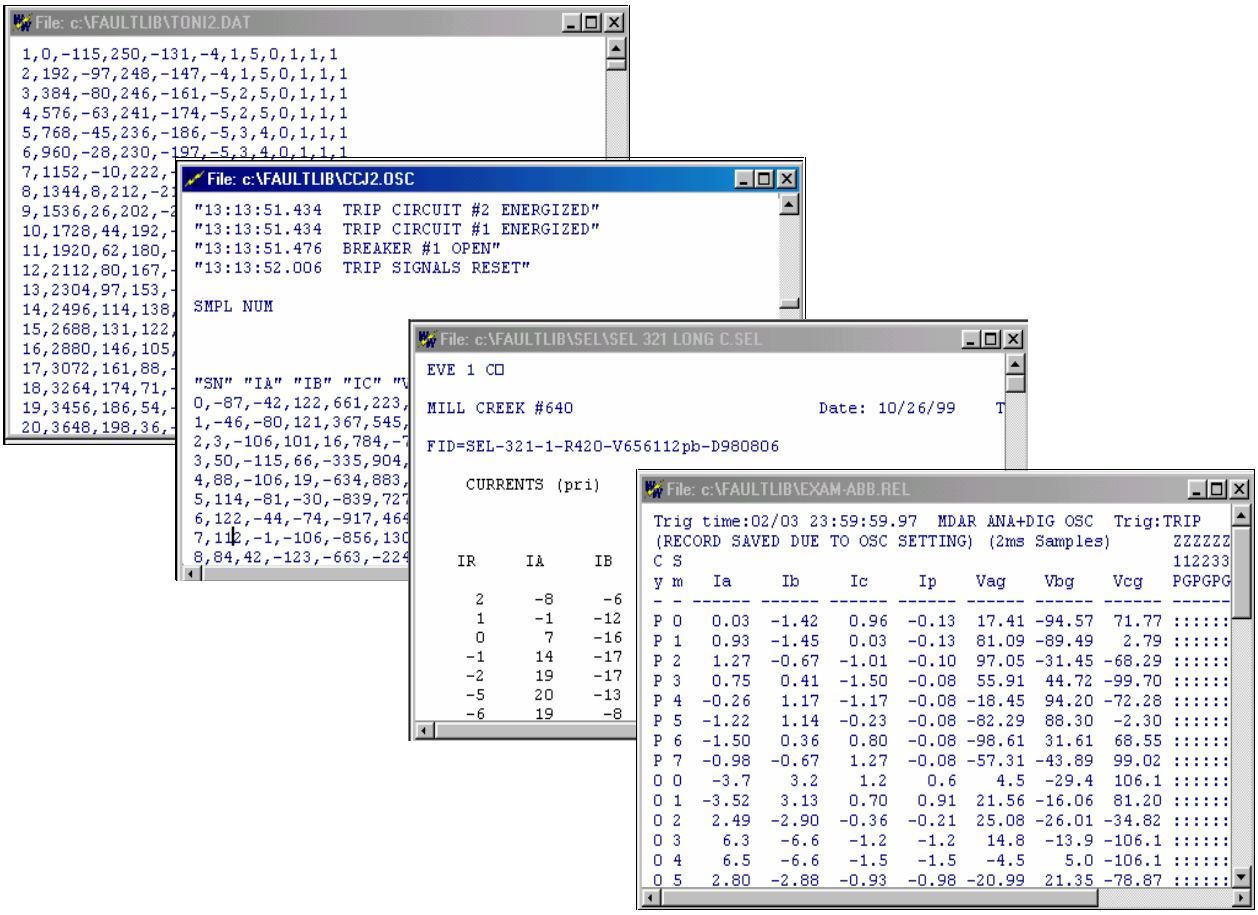

Figure-1, Various Types of ASCII Fault Record Formats

Standard: There are a few standard formats in circulation today (ASCII and Binary). The most popular one is the IEEE C37.111 Comtrade standard. Figure-2 shows an example of a Comtrade record in ASCII format, and Figure-3 shows an example of the same record but in binary format.

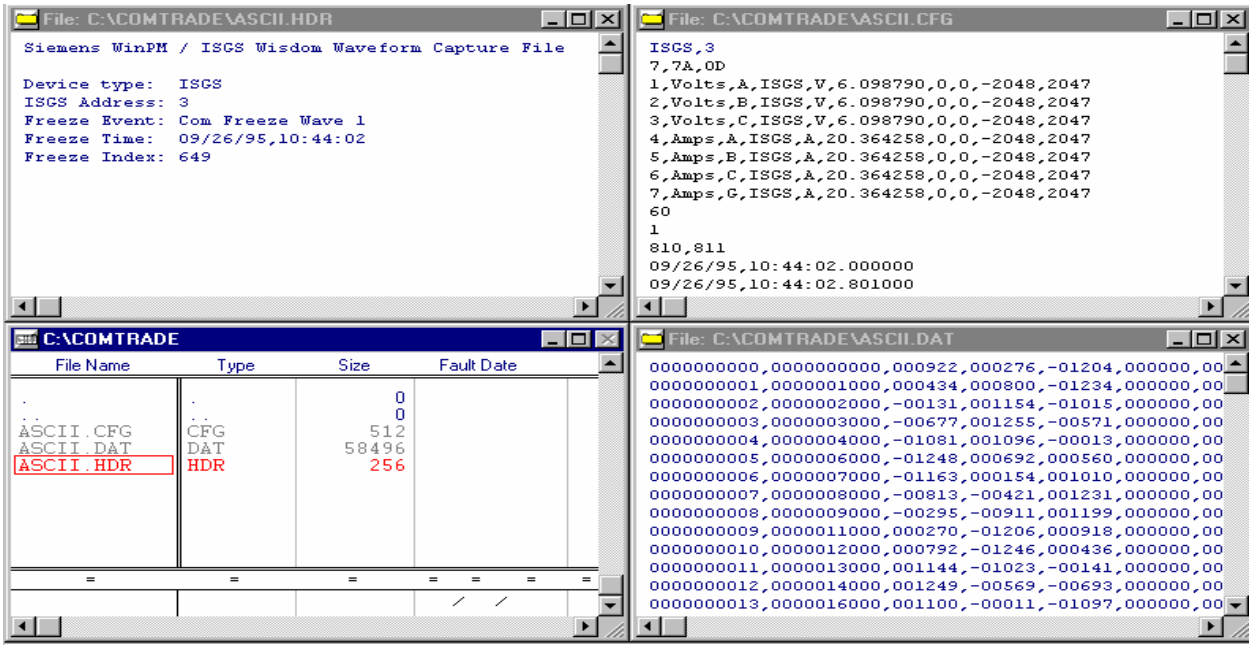

Figure-2, Standard Comtrade ASCII Format

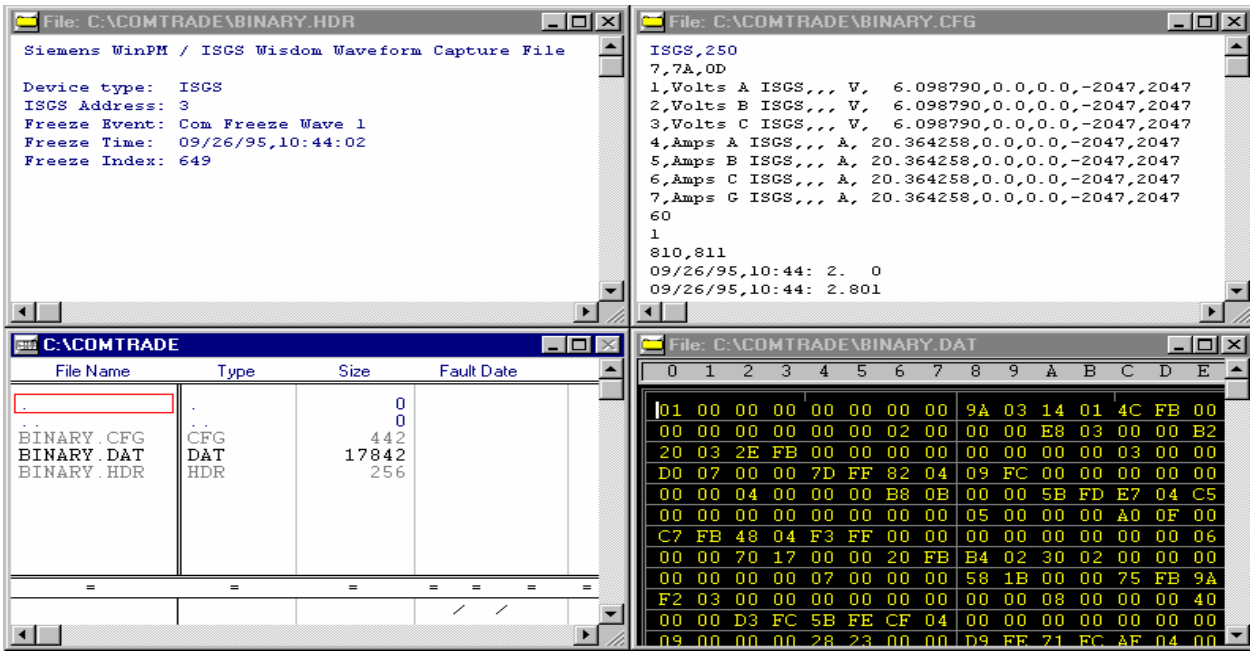

Figure-3: Standard Comtrade Binary Format

Storage: A fault record may be saved in multiple files, or multiple records may be saved in the same file. Fault records and computer files have a many to many relationship. For example, the 1991 Comtrade standard requires up to three files for each fault record: the header, the configuration and the data files as shown in Figures 2 and 3 above. The files share the same name but have different extensions: "HDR", "CFG" and "DAT". The

header file contains general information about the fault record, such as manufacturer, type and operator comments. The configuration file contains specific information, such as substation name, fault date and time, number of channels, scale factors, skew values and sampling frequencies. Finally, the data file contains the raw data samples. It is worthwhile to mention here that the header and configuration information are always in ASCII format which makes them easy to edit and change using a text editor.

On the other hand, multiple fault records may end up in the same file especially when they are collected using a terminal mode session. An example is provided in Figure-4 in the bottom right hand window.

In order to read and analyze fault records from a particular device the user must use the proprietary software that was provided with that device. Almost every device has a different type of proprietary program. Manufacturers have provided us with too many programs and operating nuances for reading fault records. This has hindered the analysis of events from various types of devices. Manufacturers have been slowly adopting Comtrade as a standard file format. Many new devices provide fault records in Comtrade allowing the use of any application that supports Comtrade to display the fault records. Because of this Comtrade is becoming a standard format for saving transient data whereas the original intent was for it to be an exchange format only. The use of Comtrade as a storage format has slowed down the use of proprietary formats and that is good news. The goal of having a universal program that can read all of the various types of data formats is close to being fully realized (see Figure-4).

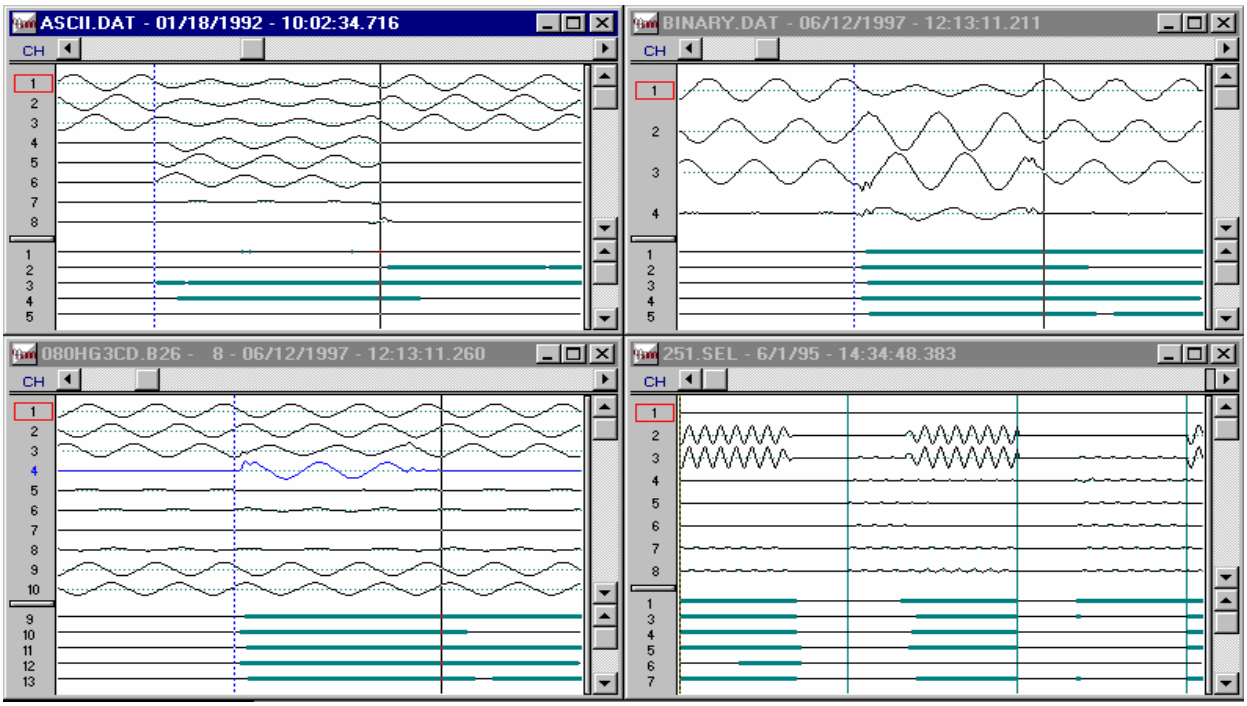

Figure-4, Common Program for Reading Various Types of Digital Fault Records

Filenames: There are many types of proprietary naming conventions in circulation today. And again, the names vary depending on the convictions of the manufacturer and the type of originating device. In general, naming conventions are organized into three (3) classes: associated, coded, and sequenced. Associated means the extension defines the type of storage format. Coded means the filename is a sequence of characters that represent specific information about the fault (such as date and time of occurrence). Sequenced means filenames are incrementally assigned.

Without having informative filenames it is impossible for any type of automated application to be easily realized and the users will have major difficulties combining and exchanging records. Uniqueness also plays a major role because filenames have to be unique otherwise they may overwrite. Realizing these problems, an IEEE-PES-PSRC working group was formed in 1999 to develop a common convention for assigning filenames that are unique and informative. The results are provided in their 2001 report titled “File Naming Convention”. This convention uses a content addressable format to uniquely tag and identify each record within a large repository. Content addressable means that the filenames contain a sufficient amount of information for users and automated analysis applications to be able to determine the extent of their contents by just reading the names. The convention is as follows and in order:

“Date, Time, Code, Station, Device, Company, User fields. Ext”

An example filename is:

“000809,175215183,-4t,Sta80,DFR717,FDA,AG-T,024km,Zone1,Trip,7kA.DAT”

There are many types of applications that can be realized given a unique, informative file naming system. Possible applications include but are not limited to time-line management, universal viewers, global fault data banks, and also automatic combining of digital fault records.

III - DESCRIPTION

So, how do we combine records from various types of devices? And also, how do we automate such a process? To answer these questions let us consider the simple case of having to combine two (2) digital fault records together. In this case, we begin by comparing the records and answering the following questions:

Source: Are the records in standard Comtrade format? “No” means we have to be able to convert or translate the formats before we can continue, while “Yes” means we can continue. Translation is essential because by definition processing routines are format independent and we need these routines to be reusable across all formats. The first step is to make sure that we have a proven translator that works with all of the various types of known or commonly used formats.

Content: Do the records have the same input channels, such as the case where one of the records is from a primary device and the other record is from the backup device and the primary and backup devices are of the same type, or such as the case where both records are from the same device? “No” means we have to *”merge”* or in other words combine by adding channels (vertical process), while “Yes” means we have to *“append”* or in other words combine by adding samples (horizontal process).

Duration: Is there a common time period between the records? In the append case: “Yes” means we can discard the common time samples from one of the records but not from both of them, while “No” means we have to invent new sample values (interpolate as needed) in order to connect the records together in time.

In the merge case: “Yes” means we can discard all of the samples that are outside the common time period from both records, while “No” means we will have to invent new samples (extrapolate as needed) in order to span both of the recorded durations on the same time scale.

Now, inventing samples is a complex science involving modeling and estimation theory and is outside the scope of this paper. Accordingly, for the purposes of this paper:

- If a common time period can not be found then we will abort without trying to append or to merge the records (combine them) because it is not practical to save records that have data files containing very large numbers of extrapolated or interpolated data samples (remember that the desired output is a single Comtrade record as described in Section I), else

- If a common time period is found then we can proceed with the next step by discarding the proper samples as described above. In the merge case the desired output is centered around the fault area containing only the common time data, and in the append case the desired output must span the total durations minus the overlapping duration.

Finding the common time period is a straight forward process of comparing time tags, but if the records are not time synchronized then the process would be a much more difficult undertaking. In that case, the samples would have to be aligned based on an event that is common to both records. For example, we can state with certainty that the sample that is at the first faulted positive peak of one record is the same point in time as the sample that is at the first faulted positive peak of the other record (provided that both records are captures of the same fault, for example the currents and voltages for one of the phases are in the first record and those for the other phases are in the second record). In manual mode we can ask the user to help out and mark the location for the first faulted positive peak in each record so we can line them up and adjust our time scales accordingly. However, in automatic mode the user is not in the loop and we will have to electronically detect the first faulted peaks by using superimposed current and/or other types of accurate algorithms for fault detection.

Sampling: Is the sampling frequency the same for both records? “No” means we have to match the frequencies before we can continue, while “Yes” means continue. If we have to match the frequencies then the first step is to select the desired rate. To avoid inventing samples and since it is easier to discard, the desired sampling rate is defined as the lowest sampling frequency among the frequencies of the digital fault records being combined.

Combining: Finally, can we combine the records? “Yes” means the formats are known and we have a common time segment and the sampling frequencies match and therefore we can merge by adding the proper columns in order or append by adding the proper rows in order. “No” means we must abort.

IV - EXAMPLES

Two examples are provided in this section and are illustrated in figures 5 thru 8. The first is an append example and the second is a merge example.

Append Example: In this example two (2) digital fault records were considered. Both records originated from the same device during the same fault. The originating device had a set limit of eleven (11) cycles per record but the fault continued beyond the eleventh cycle which caused the originating device to create a second record. Both records have exactly three (3) cycles of pre-fault data.

Figure-5 below shows the phase and residual currents from both records lined up start to end (meaning that the start of the second record follows directly after the end of the first record). A thick green line is drawn down the middle of Figure-5 to show where the first record ends and where the second one begins. The figure shows that the records did not line up properly. There is a clear discontinuity in the phase and residual traces at the green line location. This discontinuity is because of the existence of the three (3) pre-fault cycles in the second record. Figure-5 proves that we can not just append by concatenating the records, we must first discard common time. The pre-fault cycles of the second record represent a common time segment along the ending of the first record and must be discarded. Figure-6 clearly shows that once these cycles are discarded then the records do indeed line up properly. The actual fault duration as revealed by Figure-6 is about nine (9) cycles long, whereas by viewing Figure-5 the viewer could incorrectly concluded that the duration is twelve (12) cycles long. Figures 5 and 6 are evidence that we must discard before we append.

The process of discarding common time is a precision process that involves more than just delete and concatenate. In most cases there will be a small time offset (delta difference) in the sampling interval at the point of concatenation. The time tag for a sample being concatenated must be exactly one (1) sampling interval ahead of the time tag of the sample behind it. And, depending on the configuration files there could also be multiple sampling intervals. Luckily in this example both records were created by the same device and therefore the sampling intervals were consistent across them. But if

the records were created by different devices then for sure there will be a time offset between them. The offset can be calculated by subtracting the difference between the time tags at the point of concatenation from the proper sampling interval as specified in the first record's configuration file. The offset represents a small phase shift that has to be applied throughout the second record. The accuracy of the resulting alignment depends solely on the accuracy of the methods used for shifting the phase. Once the offset is applied then the sampling interval(s) of the second record can be preserved by listing them directly under the sampling interval(s) of the first record (frequency matching is not necessary in the append case).

Merge Example: In this example another two (2) records were considered. The records originated from different devices during the same fault. The devices were synchronized by the same GPS clock. The records do not share the same input channels so they can not be appended, they have to be merged. The first record contains voltage information and the second record contains current information. Both voltage and current channels are needed to calculate fault impedance and fault location. So we need to combine these records. The formats are known, there is a common time segment and the sampling frequencies are the same so the records can be merged.

Figure -7 below shows both records together, the first scan of the second record is directly underneath the first scan of the first record (a scan is a packet of samples that represent an instantaneous reading from all of the input channels). The top three (3) channels in Figure-7 are the voltages (the first record) and the remaining channels are the currents (the second record). Clearly the faulted cycles did not line up across these records and that is because we did not discard the samples that are outside the common time segment. Figure-8 shows the merged records after we discarded these samples. Note that the records are now perfectly aligned.

Here too as in the append case, we had to discard samples and that means that we will have a small time offset that has to be calculated and a corresponding phase shift that has to be applied throughout the record before we can merge. In addition and unlike the append case, frequency matching is also essential before merging because we want to avoid having gaps in the combined scans (meaning samples that are missing from certain packets). This is not to say that gaps in scans are a bad idea. Random gaps are a bad idea because they are too hard to handle, but structured and periodic gaps are a good and very useful idea because they allow us to assign a different sampling rate for each channel. The common practice today is for all input channels to be scanned at the same rate (meaning all channels are represented in each scan, no gaps). However, the Comtrade format has a feature that allows for such gaps (the value 99,999 is reserved for missing samples), and a number of devices today are taking advantage of this feature to combine high and low speed rates in the same record.

In summary, the correct sequence for merging is: match the sampling frequencies first, and then discard the samples that are outside the common time segment including calculating and applying the small time offset.

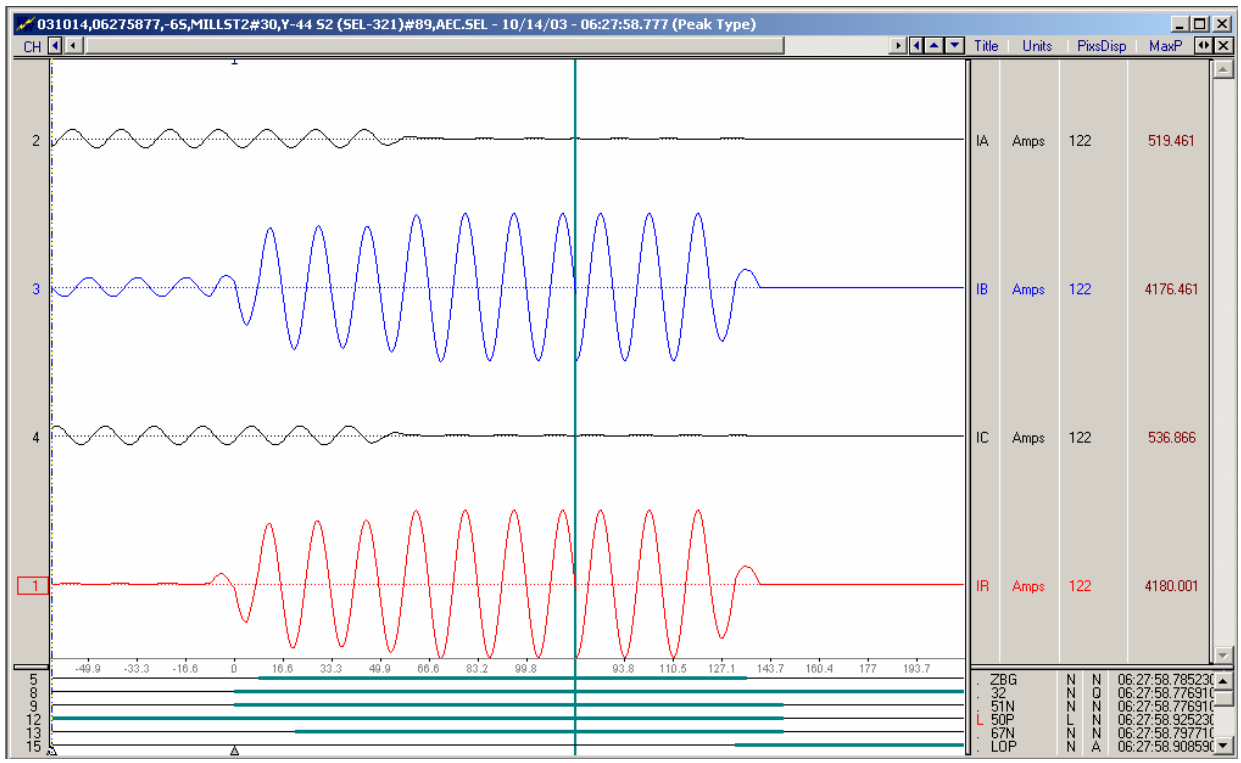

Figure-5, Append Records: Combine Horizontally (Insert New Samples at End)

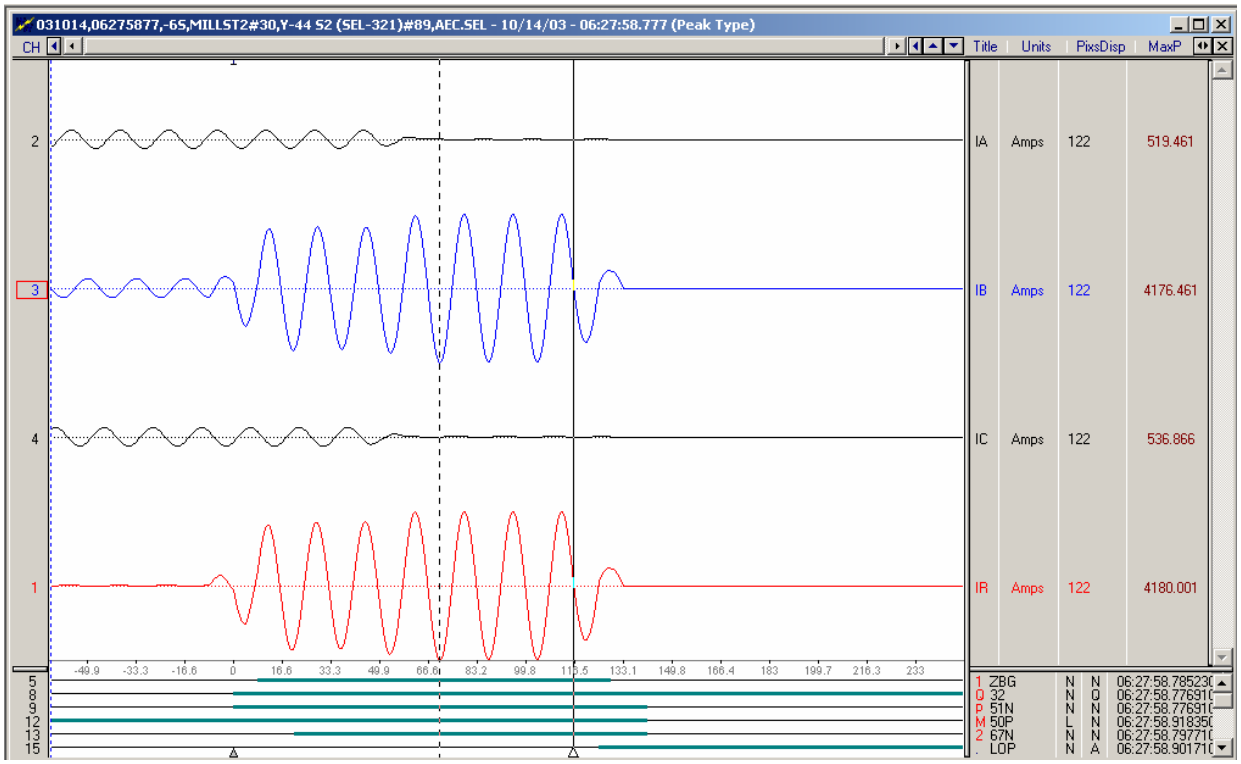

Figure-6, Append Records: Discard Common Times

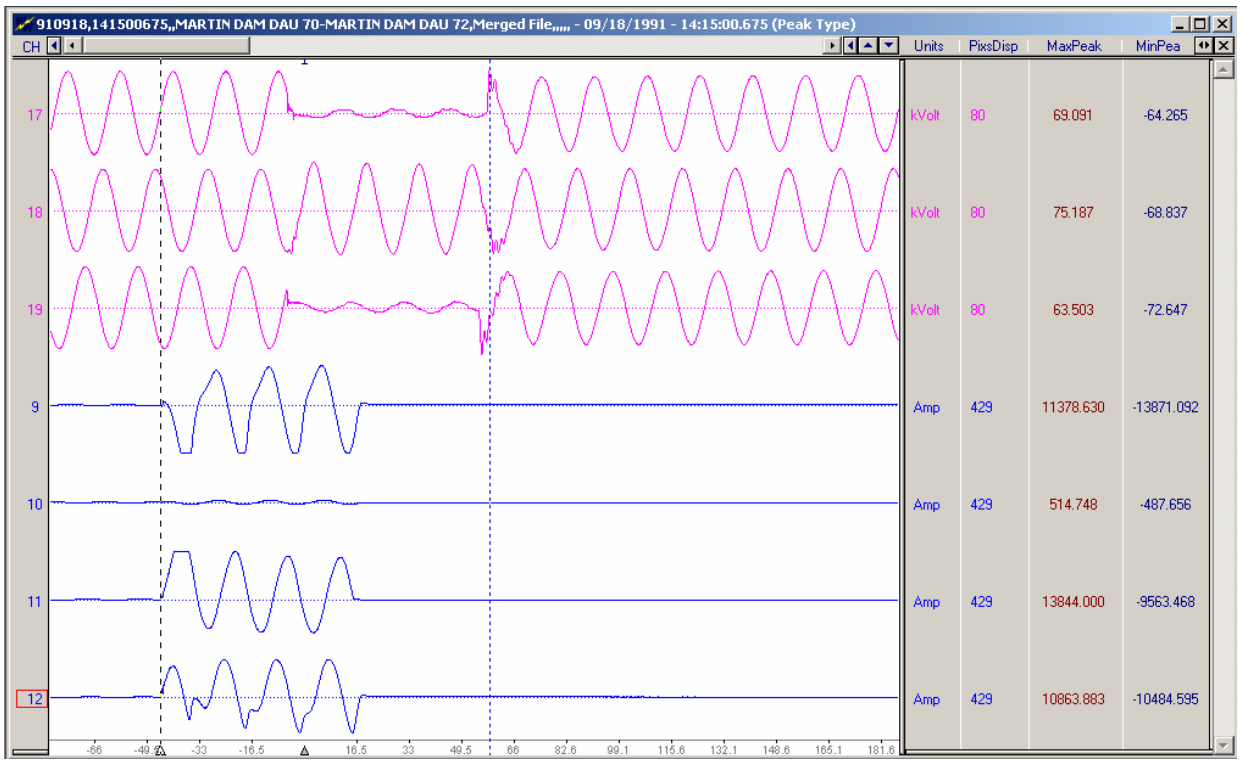

Figure-7, Merge Records: Combine Vertically (Insert New Channels at Bottom)

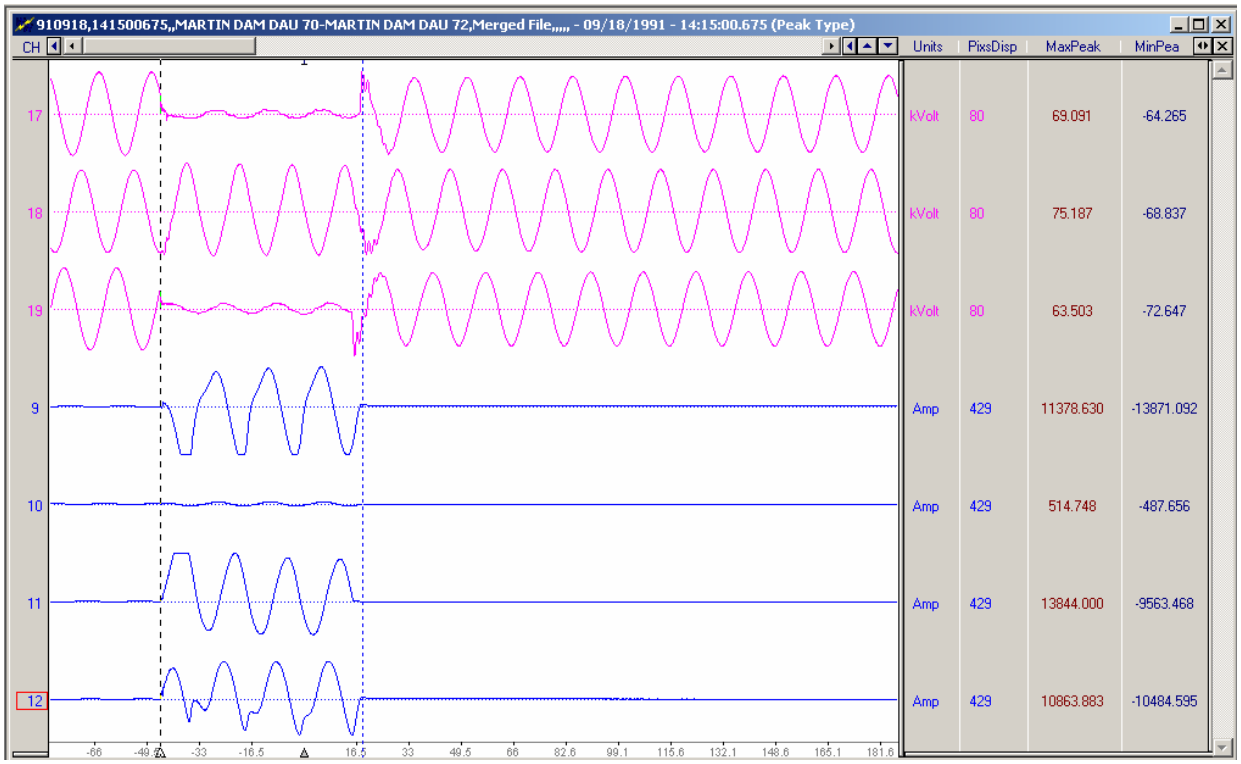

Figure-8, Merge Records: Discard Uncommon Times

V - AUTOMATION

In order to automate the process of combining records we must build a database of associations that can be used to quickly determine which channels from which records do we need to combine and when. Care must be taken when defining the main fields of this database because of the following issues:

Channels: Channels are used to monitor voltage and current information among many other things (transformer windings, polarizing potentials, temperature and humidity, relative saturation, breaker status, trip indications and so on). A DFR record may have many channels and the channel designations are usually assigned depending on the preference of the user. On the other hand, a typical record from a digital relay has fewer channels and the designations are assigned in order as specified by the originating manufacturer. Care must be taken while automatically combining records so that channel designations are not duplicated.

Associations: At least nine (9) channels are required in order to cover a three (3) phase circuit (A, B, C voltages and currents, plus IN, plus status for breakers 1 and 2). Depending on the configuration of the equipment, the data to be analyzed may be split among multiple records (as in the previous examples). In certain cases there may be a missing phase condition. For example we may have channels IA, IC, and IN defined but not IB. Here, the data for the missing phase IB can be automatically calculated by subtracting the IA and IC phases from IN, but in other cases there may be more missing phases and the subtraction will not work. Additionally, the record data could be either in primary or in secondary values and the calibration could be root mean square or peak-to-peak. Care must be taken while adjusting sample values across records so as not to misrepresent the information.

Execution: An interpreter is needed to access all of the various types of known data formats including Comtrade. The interpreter has to unpack data from the incoming records and transforms them to a form suitable for and analysis. The resulting data is to be a Comtrade record that is center around the fault area. In summary, in order to automatically combine digital fault records we will need:

- A universal reader (to reads various types of data formats and filenames)
- A common format to represent the various records in a common way (Comtrade)
- A combine engine (matches frequencies, discards samples and appends or merges)
- A universal writer (to pass results back in the original format or in another form)
- A Database (an associations knowledge-base that guides the combining process)

VI - CONCLUSIONS

Combining records is helpful but not essential for manual analysis. The users typically align their digital fault records visually and do their analysis accordingly. But when it

comes to automation, such as automatic fault location calculation, automatically sorting and combining related records is of paramount importance (it is the first step).

VII - REFERENCES

"Naming Convention for Time Sequenced Data Files," H8 working group report for the IEEE Protective Systems Relaying Committee (PSRC), Draft 4.7, April, 2005

“Event Record Sampling Rates, Record Length & Fault Analysis: Actual Examples,” Jeff Pond, Proceeding of the Fault and Disturbance Conference, Atlanta, Georgia, May 2004

"Survey of Event File Saving Schemes," A&M Makki, M. Taylor, & L. Johnson, Proceeding of the Fault and Disturbance Conference, Atlanta, Georgia, May 1999

"Survey of Event File Naming Schemes," A&M Makki, M. Taylor, Proceeding of the Fault and Disturbance Analysis Conference, Atlanta, Georgia, May 1998

"IEEE Common Format for Transient Data Exchange for Power Systems (Comtrade Standard)," IEEE Standard C37.111, 1999

"Overview of FAT, HPFS and NTFS File Systems,” Microsoft Windows NT Workstation 4.0 Resource Kit, Chapter 18, V.1997

“Object-Oriented Databases,” Ez Nahouaraii and Fred Petry, IEEE Computer Society Press, Los Alamitos, California, 1991

“Relational Databases,” Ken S. Barathwaite, McGraw-Hill, New York, 1991

“Database Principles, Programming, and Performance,” Patrick O’Neil and Elizabeth O’Neil, Academic Press, New York, 2001

“Building An Object-Oriented Database System,” Francois Bancilhon, Claude Delobel, and Paris Kanellakis, Morgan Kaufmann Publishers, San Mateo, CA, 1992