ADF File Storage Structure

for Albércorp's Battery Analysis System

This data storage mechanism is implemented using OLE2 Structured Storage. Each file will contain the following structure. Some elements of the structure may not be present in all files.

Structure

Storage "**CellCorder**" : This is the top-level storage node in the file. All Cellcorder and HydroStik data will be located within this node. Other nodes may be present at this level in future versions or applications. The CellCorder storage will contain one storage node for each individual battery whose data is contained in this file. At the present time Alber's application supports only one battery per file, but this is only an artificial limitation of the current application. It is not due to the specific design of the file system. The name of storage found directly within the CellCorder storage will represent the name of the battery whose data is contained within. The "battery name" storage will contain the following storages and streams:

Stream "**Header**" :	Contains a fixed-length structure which includes items such as the battery location, model, etc.
Stream "**Bounds**" :	Contains a fixed length structure which contains a set of high/low/baseline threshold values for each battery parameter.
Stream "**Comments**" :	Contains a variable length block of text which is edited by the user of the application.
Storage "**Data**" :	Contains one stream for each individual set of battery readings recorded on a particular date. The actual date associated with a set of readings is encoded in the name of the stream. There can be no duplicates. Each stream contains a fixed-length structure which contains all readings for all cells.
Storage "**UserDefs**" :	Contains any number of streams created by the user of the application. The name and contents of each stream is defined by the user of the application.

Data Formats

A variety of data formats, structures, and records are used to store battery information in the streams contained within the file.

Header Record : This record is found in the **Header** stream. Items of most interest to the reader are shown in **bold**.

```
cADFID: array[ 0 .. 80-1 ] of Char;
cADFType : Byte;
cADFMajorVersion : Byte;
cADFMinorVersion : Byte;

// it seems like this should be obsolete
```

```
// because the name of the storage is the name of the battery,
// but we need to keep it for backwards compatibility.
cBatteryName: array[ 0 .. MaxBatteryNameLen-1 ] of Char;

wNumCells : Word;
cTestLocation : array[ 0 .. MaxTestLocationLen-1 ] of Char;
cBatteryType : array[ 0 .. MaxBatteryTypeLen-1 ] of Char;
InstallDate : TDateTime;

wBatteryMode : Word;


// these next few things are pretty much obsolete.
// I don't think there's much we can do to salvage them because
// they are not complete, and their meaning is not clear.

wBatteryVoltage : Word;

wLowFloatVoltage : Word;
wHighFloatVoltage : Word;

wHighInternalRes : Word;
wHighInterCellRes : Word;
wCellResistance : Word;


wHighSpecificGravity : Word;
wLowSpecificGravity : Word;

// these items record the last values of the graph's manual Y axis
// min/max for each subset.

wCellVoltageYmin : Word;
wCellVoltageYmax : Word;
wCellInternalResistanceYmax : Word;
wCellInternalResistanceYmin : Word;
wCellIntercellR1Ymax : Word;
wCellIntercellR1Ymin : Word;
wCellIntercellR2Ymax : Word;
wCellIntercellR2Ymin : Word;
wCellIntercellR3Ymax : Word;
wCellIntercellR3Ymin : Word;
wCellIntercellR4Ymax : Word;
wCellIntercellR4Ymin : Word;
wCellSpecificGravityYmax : Word;
wCellSpecificGravityYmin : Word;
wCellTemperatureYmax : Word;
wCellTemperatureYmin : Word;

bTemperatureScale : Byte;
Extras:array[1..8] of cardinal;
```

Bounds Records : These records are found in the **Bounds** stream. The stream will contain one instance of a *TCellDataBounds* record, which consists of a number of *THighLowPair* records:

```
THighLowPair = packed record
        High, Baseline, Low : double;
        HighColor, BaseLineColor, NormalColor, LowColor : TColor;
        Enabled : boolean;
end;



TCellDataBounds = packed record
        Voltage:         THighLowPair;
        InternalRes:     THighLowPair;
        IntercellR1:     THighLowPair;
        IntercellR2:     THighLowPair;
        IntercellR3:     THighLowPair;
        IntercellR4:     THighLowPair;
        Temperature:     THighLowPair;
        SpecificGravity: THighLowPair;
end;
```

Cell Data Records : These records are found within each data stream within the **Data** storage node.

```
TCellRecord = packed record
    wInvalidDataFlags : Word; // not currently used

    wCellVoltage : Word;
    wCellInternalRes : Word;
    wCellInterCellR1 : Word;
    wCellInterCellR2 : Word;
    wCellInterCellR3 : Word;
    wCellInterCellR4 : Word;
    wCellSpecificGravity : Word;
    wCellTemperature : SmallInt;
 end;

TDataRecord = packed record
 // the meaning and intended usage of the ReadDate field is as follows:
 // It is valid only when the sample frame is being used outside of the context
 // of a named stream (which ordinarily represents the read date). For example,
 // ReadSampleFrame(n) will read the contents of a stream from the object and
 // at that time store the read date into the sample frame record being returned.
 // The content of the field within the stream is meaningless. Certain methods,
 // such as AddSampleFrame, use the ReadDate field as a way of passing the
 // desired read date value for the new stream. The field can be used to "remember"
 // the read date of a record when it is not possible to reference it to it's
 // original stream.
    ReadDate : TDateTime;
    Cells : array[ TCellIndex ] of TCellRecord;
    TempScale : TTempUnits;
    Extras : array[1..8] of Cardinal; // 32 bits
end;
```

Each data stream will contain one instance of a *TDataRecord*. The Cells array of the TDataRecord is currently defined as consisting of 256 instances of *TCellRecord*

Intrinsic Data types and formats

TDateTime : This is a 32-bit floating point value representing a date and time value.
Delphi stores date and time values in the TDateTime type. The integral part of a TDateTime value is the number of days that have passed since 12/30/1899. The fractional part of a TDateTime value is the time of day (as a fraction of 1 day).

Following are some examples of TDateTime values and their corresponding dates and times:

0	12/30/1899 12:00 am
2.75	1/1/1900 6:00 pm
-1.25	12/29/1899 6:00 am
35065	1/1/1996 12:00 am

To find the fractional number of days between two dates, simply subtract the two values. Likewise, to increment a date and time value by a certain fractional number of days, simply add the fractional number to the date and time value.

The fundamental integer types are **Shortint**, **Smallint**, **Longint**, **Byte**, and **Word**. Each fundamental integer type denotes a specific subset of the whole numbers, according to the following table:

Type	Range	Format
Shortint	-128..127	Signed 8-bit
Smallint	-32768..32767	Signed 16-bit
Longint	-2147483648..2147483647	Signed 32-bit
Byte	0..255	Unsigned 8-bit
Word	0..65535	Unsigned 16-bit

The generic integer types are **Integer** and **Cardinal**. The Integer type represents a generic signed integer, and the Cardinal type represents a generic unsigned integer. The actual ranges and storage formats of the Integer and Cardinal vary across different implementations of Object Pascal, but are generally the ones that result in the most efficient integer operations for the underlying CPU and operating system.

Type	Range	Format
Integer	-2147483648..2147483647	Signed 32-bit
Cardinal	0..2147483647	Unsigned 32-bit

Constants and other symbolic values

MaxBatteryNameLen = 16;
MaxTestLocationLen = 40;
MaxBatteryTypeLen = 40;
MaxDataRecords = 256;

TTempUnits : This is an enumerated type used to represent various temperature scales such as Celsius, Fahrenheit, and Kelvin. Although the ADF file maintains some notion of differing temperature scales, all temperature information is stored internally in degrees Celsius.