NOTES TO SELF: Add S-64 test from S-64.

See <https://www.iho.int/iho_pubs/standard/S-64/S-64_Edition_3.0.1/S-64%20Edition%203.0.1.pdf>

These are some rough notes on using the S57 file format with SharpMap and GDAL/OGR. The existing documentation on this is either fairly sparse or incredibly complex. If you want to get in to the gory details then try to read the IHO documentation. It is much more focused on how to generate the file than it is on how to use it. By far, I got more out of closely examining the sample code and documentation from the GDAL/OGR and SharpMap distribution than I did from reading all that stuff initially. I say initially because as you start to grasp what the documentation is saying, then it is nice to fill in some of the details with the IHO documentation.

Before we start down that path, I need to make a very important point. There are a lot of overlaps between SharpMap and GDAL/OGR. While SharpMap is built on top of GDAL/OGR it does not encapsulate every feature of GDAL/OGR. In the cases where it does, the methods used often produce different results with the same name or the methods are incompatible. This is made worse by the lack of any API documentation for SharpMap or for that matter any comments within the code. So, if you are looking for how to read the features of a layer, it is easy to find in GDAL/OGR, but not so easy within SharpMap.

Two other points worth noting about SharpMap. I don’t know the status or future of that project. What I do know is that is using very old versions of almost all of its dependent projects. For example, it is using GDAL/ORG version 1.1 while the current version is 2.1. I should note that GDA/ORG seems very much a live and vibrant project. SharpMap, on the other hand, doesn’t. It appears to go months between any contributions and the development seems sporadic at best. Getting questions answered requires a lot of patience and the answers are often terse.

The other point is that SharpMap uses the C# automatically generated shell over GDAL/OGR so the methods implemented are not always exactly the same as the GDAL/OGR documentation indicates, especially since the SharpMap version is quite old. To make this worse, the SWIG generated shell is not the friendliest environment for the Visual Studio IDE. For example, if we sit a breakpoint and look at a “Feature” class, all we see is the SWIG construct for it, not the actual data it contains. For this reason, my code has a lot of debug code that prints to the console or a file so that I can see what is going on. For me, this method is a lot faster than using the IDE for most of the development.

One might ask, why use SharpMap at all. The answer to that is it does things that GDAL/OGR does not do and it allows us to do those from C#. I am not a fan of C++ and was a very early adapter of C# and it is very much my language of choice. In addition, Microsoft no longer supports Windows Form development in C++, and while there are work-arounds for that (none that I have ever been successful with I might add), I find it much easier to write Windows applications in C#. So, I live with the wrinkles that come with SharpMap. The only other choice seems to be DotSpatial and I found it much harder to use. While it does have documentation it seems very out of date. More importantly to me, it does not support S-57 data directly, but you have to add and extension manager then add GDAL/OGR. I got that far, had all kinds of issues getting it to work and finally gave up.

If you don’t read anything else about GDAL/ORG read this page: <http://www.gdal.org/ogr_arch.html>

It may be something you have read several times or come back to several times, as I have, but it contains a nice summary of how OGR works and a bit about why some of it works the way it does.

The second stop should be <http://www.gdal.org/osr_tutorial.html> which is just about everything one needs to know about coordinate transform systems. Note that WGS\_1984 seems to be the most common system.

The Geospatial world seems to love acronyms. Two of the most common are WKT (wkt) which stands for Well Known Text. WKT is nothing more than a standard way to write coordinates so that other systems can understand them without translating the format. The other is WKB (wkb) which stands for Well Known Binary and is basically the same as WKT except that instead of being the ASCII representation the data is left in its binary form. Both formats are scattered throughout GDAL/OGR and SharpMap.

In brief, a S57 file consists of a number of layers. Layers are an abstract concept that ties together everything about a similar set of objects. For example the layer called BCNLAT contains all the information of about lateral buoys while SOUNDG contains all the sounding data.

Some of the layers have displayable and some do not. The non-displayable layers typically contain information about the map itself, while the displayable layers contain references to object. It has to be noted that the S57 file contains no directly displayable information. It contains objects that must be translated into their appropriate display objects and color.

For each layer in the S-57 file, there is a corresponding Feature Definition. There is a one to one relationship between these two although in the OGR implementation they exist in parallel. The FeatureDefn tells everything that is common to all elements of that layer or feature. From the

The basic structure of each layer is exactly the same. The first set of elements is the "Feature Definitions". Each "Feature Definition" has a name.

This dataset tells us how to interpret the next layer. For example, it contains data such as what data type each element, how many digits it contains, and what precision it has.

For example, the first layer that is typically in each map is DSID. This is a non-displayable layer, but contains all the pertinent information about the map. Here are the first few entries of this layer from a file I dumped.

DSID\_EXPP: Integer (3.0)

DSID\_INTU: Integer (3.0)

DSID\_DSNM: String (0.0)

DSID\_EDTN: String (0.0)

DSID\_UPDN: String (0.0)

DSID\_UADT: String (8.0)

DSID\_ISDT: String (8.0)

This shows the first few entries. EXPP is an integer with three digits with no digits to the right of the decimal point. Some of these seemingly make little sense. For example DSNM is a String with 0 digits, but as we will see, that is not the case.

There is only one Feature Definition dataset for each layer.

The next dataset is called “Features”. The elements have exactly the same names as the “Feature Definitions”. There is at least one “Feature” dataset, but there may be several. In all cases, they match the names of the “Feature Definitions”.

Field Name: DSID\_EXPP, Field Name Ref: DSID\_EXPP, Field Type Name: (Integer, Field Type: OFTInteger, Value: (1)

Field Name: DSID\_INTU, Field Name Ref: DSID\_INTU, Field Type Name: (Integer, Field Type: OFTInteger, Value: (5)

Field Name: DSID\_DSNM, Field Name Ref: DSID\_DSNM, Field Type Name: (String, Field Type: OFTString, Value: (US5TX51M.000)

Field Name: DSID\_EDTN, Field Name Ref: DSID\_EDTN, Field Type Name: (String, Field Type: OFTString, Value: (1)

Field Name: DSID\_UPDN, Field Name Ref: DSID\_UPDN, Field Type Name: (String, Field Type: OFTString, Value: (0)

Field Name: DSID\_UADT, Field Name Ref: DSID\_UADT, Field Type Name: (String, Field Type: OFTString, Value: (20010711)

Field Name: DSID\_ISDT, Field Name Ref: DSID\_ISDT, Field Type Name: (String, Field Type: OFTString, Value: (20010711)

Notice that the names match exactly. In the case of the features, the actual values are included. To decipher these values, we need to use the S57 magic decoder. The two I know of are   
<http://www.s-57.com/> and <http://www.caris.com/s-57/frames/S57catalog.htm> . I use the second one almost exclusively as it seems easier to get around in. In both, the left frame shows the Object class, which corresponds to the layer and the Attributes with the layer show up on the right side. These correspond to features.

So, if we the look at the Object Class ACHARE (Anchorage Area), we see that it has three levels of attributes. Most of these are of no interest to us, but a few are. Also notice that none of these are required fields. To me, this seems odd. Notice that none of these Attributes specific the location of this object. The location of the object is contained in one of two places. Each field can have a geometry entry which will specify its location is the layer contains only a single object. It may have sub-geometries (many in fact) if there are multiple objects in the feature.

If we look at the object type BCNLAT, we see it contains several mandatory fields, including the shape, category, and color or color pattern. If we click on CATLAM (category) we see how the various buoy categories are encoded. Click on BCNSHP and we can see how the various buoy shapes are encoded. This is a good time to again mention that the S-57 specification does not tell us how to render these, but only supplies the information needed to render these.

There are several object classes are worth noting as starting points. Each object name has a corresponding object type, which is often easier to use. The type is shown in parenthesis next to the name. In OGR, the OBJL field represents this type.

BCNCAR (5) are prominent objects used as landmarks. They are often named, have required shapes and colors.

BCNLAT (7) represent lateral buoys. These are often named and have required shapes.

BCNSAW (8) represents safe water buoys and have shapes and colors.

BOY???? (15-19) represents permanent buoys. There are several objects in this category. They have required shapes and colors and are often named.

BRIDGE (11) represents bridges. CATBRG is required and indicates the type of bridge. VERCCL, VERCLR, VERCOP, and COLPAT are all required. They may and often do have HORCLR as well.

CBLOHD (21) represents an overhead cable. VERCLR and VERCSA are required fields

COALNE represents the coastline boundary. One nice thing about [WWW.S-57.COM](http://WWW.S-57.COM) is that in COALNE if you click on CATCOA, then in the right panel mouse over the INT 1 categories, is shows the graphical interpretation.

LNDARE is a bit of a misnomer. These seem to represent features built on land that extend into the water.

SEAARE is the sea area within the defined name, which is a required field. It includes things like pinnacles, mudflats, basins, shoals, and other things a seaman might like to know about.

OBSTRN are any obstructions to navigation. They can be natural or manmade. Required fields include VALSOU which is the value of sounding that represents the depth of the object (negative numbers indicate that it is out of the water) as well as WATLEV which describes weather the object is submerged or not.

If we go to that site, we will not find a layer called DSID. That is because, as best I can tell, s-57.com only includes information about displayable layers. DSID is not a displayable layer, but contains meta-data about the map.