**S-101PT12-06.11 Rev1**

# Paper for Consideration by S-101PT

# Data Display Algorithm

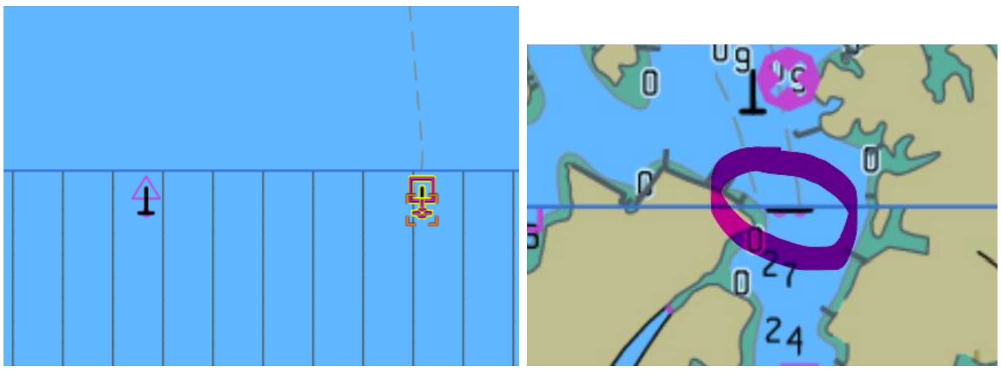
| | |
|---|---|
| ***Submitted by:*** | Scales Sub-Group Lead |
| ***Executive Summary:*** | This paper proposes the inclusion of a Data Display Algorithm in S-101 Edition 2.0.0. |
| ***Related Documents:*** | S-101 2.0.0 |
| ***Related Projects:*** | S-101 |

## Introduction / Background

1. Some implementers have expressed a need to incorporate algorithms in S-101 PS to clarify the dataset selection and data display on the ECDIS graphics window.
2. S-101 PS 1.2.0 contains an Annex D - Dataset Loading Algorithm:
    - D-1 Scale Bands
    - D-2 Dataset Coverage Selection Process
    - D-3 Data Display Algorithm: To be defined

## Analysis/Discussion

3. A data rendering algorithm (annexed to this paper) has been proposed by SevenCs prior to S-101PT11 but was not included into S-101 Ed. 1.2.0.
4. Issue https://github.com/iho-ohi/S-101-Documentation-and-FC/issues/71 is currently open in which NIWC identifies problems when drawing datasets with different minimum display scales.



*"The two beacons are obscured at small scale because the datasets are stacked rather than interleaved"*

5. As a solution, NIWC suggests encoding a new portrayal attribute on Data Coverage features.
6. In absence of feedback from other OEMs, it is difficult to have a definitive opinion. Alternative solutions and further testing are necessary to validate the best solution.

## Recommendations

It is recommended to:

- Add the data display algorithm provided by SevenCs in Annex D;
- Carry one with testing (OEMs are kindly encouraged to provide feedback).

**Action Required of S-101PT**

The S-101PT is invited to:

a) Discuss this paper;
b) Agree on the recommendation.

# Annex: Data Display Algorithm

## D3 Data Display Algorithm

### General

After the data-coverages are selected and the associated data-sets are loaded the chart display will be generated by:

1. Create a set of drawing instructions for each dataset. This step is called portrayal and defined by the rules in the portrayal catalogue.
2. Render the drawing instructions as described below.

Notes:

- Datasets can only be portrayed entirely, there is no mechanism to only portray single data-coverages
- The algorithm assumes that the rendering is made by using a kind of the 'Painters algorithm'. This means an opaque fill will completely obscure what has been rendered at this position before. This does not mean that any implementation must follow this approach, other techniques like Z-Buffer technique may be used. The algorithm will not give implementation details, any implementor has the freedom to reach the desired result in the most effective way.

### The Rendering Algorithm

The first step is to group the datasets into subsets which we will denote 'Layer'. The criteria for the separation will be the minimum display scale of the dataset. Note that all data-coverages within a dataset will share the same minimum display scale and data-sets with the same minimum display scale are not allowed to overlap. To be precise, the union of all data-coverages of one dataset must not overlap the union of the data-coverages of another dataset with the same minimum display scale.

Then the 'Layers' are sorted by their minimum display scale and sequentially rendered starting with the smallest minimum display scale.

**Algorithm**: *RenderChartImage*

**Input**: A set of datasets *dataSets*
A drawing device

1. Split the set dataSets into sub-sets denoted l*ayer*$_0$, *layer*$_1$, … such that the minimum display scale of each dataset in one *layer*$_x$ is the same.
2. Sort the *layer*$_1$ .. *layer*$_n$ by its associated minimum display scale
3. Clear the drawing device (e.g. by filling the drawing device with the NODTA colour or pattern.
4. Iterate over all *layer*$_x$ starting with the smallest minimum display scale
5. Render the layer with the algorithm *RenderLayer*

Note: For the sake of simplicity the concept of display planes (i.e. under and over radar) is not considered here. Without loss of generality the algorithm can be used multiple times to create the images for each display plane. One way of achieving it is to split the output of the portrayal into subsets one for each display plane and run the algorithm for each subset.

Note: The algorithm as described here does not distinguish between official and non-official data. It could be achieved by taking this into account during the grouping of the input datasets.

## The Algorithm RenderLayer

This algorithm describes how the datasets of one layer i.e. those that have the same minimum display scales are rendered.

```
Algorithm: RenderLayer
Input:  A set of datasets dataSets that have the same minimum display scale
        A drawing device
  1. For each display priority displayPriority starting with the smallest
        a. Collect the drawing instructions (except text instructions) from each dataset's display
           instructions that are assigned to displayPriority
        b. Render the area instructions from that collection
        c. Render the line instructions from that collection
        d. Render the point instructions from that collection
  2. For each display priority displayPriority starting with the largest
        a. Collect the text instructions from each dataset's display instructions that are assigned to
           displayPriority
        b. Render the text instructions
```

Notes:

- 1b, 1c, 1d, and 2b:
  Rendering must take the *viewingGroup*, *scaleMinimum*, and *scaleMaximum* properties of the display instruction into account. (See S-100 9-11.2)
- 2b:
  Optionally, an implementation may check if the text to be rendered will overlap any text already drawn. In that case the text will not be drawn. With the reversed order of the display priorities texts with a higher priority will be visible. Drawing both texts will make them unreadable.