**Floating Weather Station Module Guide\*[[1]](#footnote-2)**

This guide to the modules of the Floating Weather Station Family is patterned after the A-7E Module Guide [6] and uses the structuring principles describe therein and in [41]. Accordingly, we quote, with suitable emendations, including the substitution of FWS for A-7, a section of the A-7 module guide to describe further the purpose of the FWS module guide.

The FWS module guide provides an orientation for software engineers who are new to the FWS family, explains the principles used to design the structure, and shows how responsibilities are allocated among the major modules.

This guide is intended to lead a reader to the module that deals with a particular aspect of the system. It states the criteria used to assign a particular responsibility to a module and arranges the modules in such a way that a reader can find the information relevant to his purpose without searching through unrelated documentation.

This guide describes and prescribes the module structure. Changes in the structure will be promulgated as changes to this document. Changes are not official until they appear in that form. This guide is a rationalization of the structure, not a description of the design process that led to it.

Each module consists of a group of closely related programs. The module structure is the decomposition of the program into modules and the assumptions that the team responsible for each module is allowed to make about the other modules.

**Goals of the Module Structure**

The overall goal of the decomposition into modules is reduction of software cost by allowing modules to be designed, implemented, and revised independently. Specific goals of the module decomposition are:

(1) each module’s structure should be simple enough that it can be understood fully;

(2) it should be possible to change the implementation of one module without knowledge of the implementation of other modules and without affecting the behavior of other modules;

(3) the ease of making a change in the design should bear a reasonable relationship to the likelihood of the change being needed; it should be possible to make likely changes without changing any module interfaces; less likely changes may involve interface changes, but only for modules that are small and not widely used. Only very unlikely changes should require changes in the interfaces of widely used modules. There should be few widely used interfaces;

(4) it should be possible to make a major software change as a set of independent changes to individual modules, i.e., except for interface changes, programmers changing the individual modules should not need to communicate. If the interfaces of the modules are not revised, it should be possible to run and test any combination of old and new module versions.

As a consequence of the goals above, the FWS software is composed of many small modules. They have been organized into a tree-structured hierarchy; each non-terminal node in the tree represents a module that is composed of the modules represented by its descendents. The hierarchy is intended to achieve the following additional goals:

(5) A software engineer should be able to understand the responsibility of a module without understanding the module's internal design.

(6) A reader with a well-defined concern should easily be able to identify the relevant modules without studying irrelevant modules. This implies that the reader be able to distinguish relevant modules from irrelevant modules without looking at their internal structure.

## Design Principle

The FWS module structure is based on the decomposition criteria known as information hiding [35]. According to this principle, system details that are likely to change independently should be the secrets of separate modules; the only assumptions that should appear in the interfaces between modules are those that are considered unlikely to change. Every data structure is private to one module; it may be directly accessed by one or more programs within the module but not by programs outside the module. Any other program that requires information stored in a module's data structures must obtain it by calling module programs.

Applying this principle is not always easy. It is an attempt to minimize the expected cost of software and requires that the designer estimate the likelihood of changes. Many of the changes that are accommodated by the module structure described in this document are guided by the variabilities described in the Floating Weather Station Commonality Analysis shown in HGURE 33.

## Module Description

This document describes the module structure by characterizing each module's secrets. Where useful, we also include a brief description of the services provided by the module. Where a module's secret is directly concerned with a variability from the commonality analysis, we also identify the variability.

The remainder of this document consists of two parts:

* a top-down overview of the module structure, and
* a graphical depiction of the module structure.

**1. Behavior Hiding Modules**

The behavior hiding modules include programs that need to be changed if the required outputs from a FWS and the conditions under which they are produced are changed. Its secret is when (under what conditions) to produce which outputs. Programs in the behavior hiding module use programs in the Device Interface module to produce outputs and to read inputs.

### 1.1. Controller

Service

Provide the main program that initializes a FWS.

Secret

How to use services provided by other modules to start and maintain the proper operation of a FWS.

Associated variabilities and parameters of variation

None

### 1.2. SOS Controller

Service

Provide the main program that initializes a FWS.

Secret

How to maintain the proper operation of a FWS in SOS mode.

Associated variabilities and parameters of variation

V4. SOS support; SOSEquipment

### 1.3. Message Generation

Service

Periodically retrieve data from the Data Banker and transmit it.

Secret

How to use services provided by other modules to obtain averaged wind speed data and trans­mit it at a fixed period.

Associated variabilities and parameters of variation

TransmitPeriod, SosTransmitionValue

### 1.4. Message Format

Service

Support construction of an output message.

Secret

How to create a message in the correct format for transmission.

Associated variabilities and parameters of variation

MsgType

## 2. Device Interface Modules

The device interface modules consist of those programs that need to be changed if the hardware devices to FWSs or the output to hardware devices from FWSs change. The

input from secret of the device interface modules is the interfaces between FWSs and the devices that produce its inputs and that use its output.

### 2.1. Sensor Device Driver

Service

Provide access to the wind speed sensors. There may be a submodule for each sensor type.

Secret

How to communicate with, e.g., read values from, the sensor hardware.

Associated variabilities and parameters of variation

V7, the wind speed sensor hardware.

Note

This module hides the boundary between the FWS domain and the sensor domain. The boundary is formed by an abstract interface that is a standard for all wind speed sensors. Pro­grams in this module use the abstract interface to read the values from the sensors.

### 2.2. Transmitter Device Driver

Service

Provide access to the transmitter.

Secret

The details of the transmitter hardware.

Associated variabilities and parameters of variation

V8. the transmitter hardware.

Note

This module hides the boundary between the FWS domain and the radio transmission domain. The boundary is formed by an abstract interface that is a standard for all radio transmitters. Programs in this module use the abstract interface to send messages to the transmitter to be broadcast.

### 2.3. Button driver

Service

Provide access to a hardware button.

Secret

The details of the hardware button.

Associated variabilities and parameters of variation

V10. The SOS and reset buttons.

### 2.4. Light driver

Service

Provide access to a light.

Secret

The details of the light hardware.

Associated variabilities and parameters of variation

V10. the SOS light.

## 3. Software Design Hiding Modules

The software design hiding modules hide software design decisions based upon programming considerations such as algorithmic efficiency. Both the secrets and the interfaces to this module are determined by software designers. Changes in these modules are more likely to be motivated by a desire to improve performance than by externally imposed changes.

### 3.1. Sensor Monitor

Service

Periodically retrieve data from the wind speed sensor(s) and deposit it in the Data Banker.

Secret

How to use the services provided by other modules to obtain wind speed data at a fixed period and store it for later retrieval.

Associated variabilities and parameters of variation

SensorCount and SensorPeriod

### 3.2. Data Banker

Service

Store the most recent wind data.

Secret

The algorithm and data structure used.

Associated variabilities and parameters of variation

None

### 3.3. Averager

Service

Process the current Data Banker data to produce a current wind speed estimate.

Secret

The algorithm used.

Associated variabilities and parameters of variation

LowResWeight, HighResWeight, and History

## 4. Issues

Issue 1: Should the decisions of how often to read sensors and how many sensors there are be hidden in one module?

Al: No. These are two distinct parameters of variation that can change indepen­dently and as independent decisions they should be hidden in different modules. In this structure we would have a sensor monitor module and a sensor reader module. (There could be one instance of the sensor reader module per sensor.) The sensor monitor module's responsibility is to inform the sensor reader mod­ule(s) what the SensorPeriod is. This could be done by the use of a SENSORPE­RIOD constant supplied at compile time to different instances of the sensor reader modules. Each instance could be implemented as a process. Each sensor reader module is then responsible for walling up at the specified period, reading its sensor, and storing its reading in the data banker for later retrieval. The sensor monitor module then becomes a module that runs at or before compile time.

A2: Yes. The decisions are not as independent as they appear. Because of timing constraints, adding sensors means that the program may not have time to read sensors as often. In this structure we would have a sensor monitor module and a sensor reader module. (There could be one instance of the sensor reader module per sensor.) The sensor monitor module's responsibility is to obtain sensor data from the sensor reader modules. The sensor reader module's responsibility is to provide sensor data when it is invoked.

**Resolution:**

Alternative Al: would give us more independence in changing the val­ues for SensorCount and SensorPeriod, except that we would still have to recalculate the system timing when we changed either. It might require slightly less run-time code, and would probably increase the amount of process switching overhead. Since SensorCount and Sen­sorPeriod are not as independent of each other as we would like because of their joint effect on system timing, and since alternative A2 may require less process switching, we will choose A2.

Controller

Behavior

Hiding Message Generation (TransmitPeriod P13)

Message Format (MsgType P4)

Sensor Device Driver (Sensor hardware V7)

FWS Device

Interface Transmitter Device Driver (Transmitter hardware V8.)

Sensor Monitor (SensorCount Pl0, SensorPeriod P11)

Software

Design Hiding Data Banker

Averager (HighResweight P1,

LowResWeight P2, History P3)

FIGURE 4-24. Floating Weather Station Module Hierarchy with Mapping to Parameters of

Variation and Variabilities

1. The FWS Module Guide is taken, with permission, from Engineering Domains: A Family Based Software Development Process, David M. Weiss and Chi Tau Robert Lai, Addison Wesley, in publication. [↑](#footnote-ref-2)