The Fuel System of a Plane.

Introduction:

The system demonstrated a simple fuel gauge readings in one plane that has been tasked to fly three times in different parts of the world and any flight captain will need to ensure that their readings are correct otherwise the plane will not fly the three flights that it has been tasked too.

1. Structure

The fuel system is developed where it fits the specification for a plane and the fuel revolves around whether it is on or off whenever the fuel system in the plane is ready to be running on the flight and can handle the flight depending on if there is enough fuel for the designated continents. For example, if the system is off, then there is not enough fuel to fly for the plane for each flight as this system utilizes only one plane. (This can be seen in the runs pdf file)

The code has multiple procedures where it suggests a function converting an integer to a string which allows to get an output that satisfies the systems which lets the plane fly if an integer has a correct input that shows in the display procedure. For example, in listing 1 the code illustrates the conversion to have the program where an integer returns as a String.

function SFSTS (SFS: Status_Fuel_T) return String; *(1)*

2. Data structure

The data structure within the systems ensures the system works effectively, this can be shown in my integers where there are multiple flights each having their own designated fuel gauges tanks and having their own outputs. The constant integers signify the continents of each flight that the plane is designated to be flying and this can be shown in the types representing the fuel range for different flight segments. Moreover, an added type of record was necessary to utilizes in post conditions, dependency clauses and in global variables such as 'FFS' and 'SST'. A record type suggests that it will allow to store the record and will be able to keep record type in one variable that will allow them to create the fuel status (ads file in line 40). Furthermore, the system includes enumeration, which is the main feature of the system as this allows us to ensure if the system is safe or not.

2.1 Loop Invariant

The Invariant rules can be specified as functions in Ada SPARK and constitute conditions that will constantly appear in every single procedure in the system. By translating this predefined set of rules into a logical language, we obtained a set of specifications which when combined the given loop invariant for the system.

The invariant loop in this system creates the printing, the measured fuel status and how much fuel in the tank is there within the plane in the main file when running an application.

```
pragma Loop_Invariant (ISS(FFS));
   RF;
   MFS;
   PS;
```

```
end loop;                                                    (2)
```

3. **Procedure and functions involving (pre and post conditions)**

The current fuel system has multiple procedures and functions that allow the system to work and be safe. For example, the necessary post conditions within the system are to ensure the outputs of the whilst the dependency clauses suggest that the inputs and the outputs by initializing and printing those inputs and outputs when the system is running that is shown in listing (4)(5).

The function of the safety system is seen below in listing (3) that demonstrated is main initiator of the system where the integer is greater than 'FRC' then the output of the fuel system will be 'ON' thus allows the plane to fly however there are other examples where it shows that the integers are supposed to be inputted in their corrected values and if not then the system will be unsafe if the outputs shows 'OFF'.

```
function ISS (Status : SST) return Boolean is ---
    (if Integer(Status.FM) > FRC
     then Status.SFS = ON
     else Status.SFS= OFF);                                  (3)
  procedure Init with
    Global => (Output => (Standard_Output,Standard_Input, FFS)),
    Depends => ((Standard_Output,Standard_Input, FFS) => null),
    Post    => ISS(FFS);                                     (4)
 procedure PS with
    Global => (In_Out => Standard Output,
               Input  =>  FFS),
    Depends => (Standard Output => (Standard Output, FFS));  (5)
```

Conclusion

In conclusion, this system includes the necessary procedures and functions to utilize the fuel system handling the safety of the system based on the inputted fuel values which simultaneously the functionality of pre-condition and post conditions are a necessity to control this plane fuel safety system.