# GOTO: FILE -> MAKE A COPY to save for your own use.

# 

# Section 1 - Project Description

## 1.1 Project

Elevate Oil Level Sensor Test Tank

## 1.2 Description

The Elevate Oil Level Sensor Test Tank (Elevate OLS-TT) uses LabVIEW to Automate the ATP process and add AEPS Testing functionality.

## 1.3 Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Comment** | **Author** |
| 6-10-2024 | Initial Release | R. Ales Consulting |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Contents**

[Section 1 - Project Description](#_yr1n1w74g294)

[1.1 Project](#_mpniepv3sji5)

[1.2 Description](#_kxtdscgtj3ad)

[1.3 Revision History](#_x1jfzxbgcdpj)

[Section 2 - Overview](#_v0z2li3lbhgp)

[2.1 Purpose](#_wkcniuopwsk)

[2.2 Scope](#_40qz41btxs84)

[2.3 Requirements](#_b5joi8kutkvr)

[2.3.1 Estimates](#_p6lz0c1jpkfa)

[2.3.2 Traceability Matrix](#_khayjwa8jo2y)

[Section 3 - System Architecture](#_yeyfpufiww8s)

[Section 4 - Data Dictionary](#_drq5ibko1acc)

[Section 5 - Software Domain Design](#_mc3s4e33quku)

[5.1 Software Application Domain Chart](#_hzjgbuitbc87)

[5.2 Software Application Domain](#_mb7tg0hm62qt)

[5.2.1 Domain X](#_1bzzrr7eu9b7)

[5.2.1.1 Component Y of Domain X](#_qt8nkuvo35yn)

[5.2.1.1.1 Task Z of Component Y1 of Domain X](#_av8bqurqa2qq)

[Section 6 – Data Design](#_wk232hcifrl1)

[6.1 Persistent/Static Data](#_quga8kcr4qx9)

[6.1.1 Dataset](#_2rfx518fnjz5)

[6.1.2 Static Data](#_9as0jpuz4i6o)

[6.1.3 Persisted data](#_ktmhbqm32m9h)

[6.2 Transient/Dynamic Data](#_kfqngvi55nuv)

[6.3 External Interface Data](#_s8ifm2hzhj88)

[6.4 Transformation of Data](#_39glqs2tdqgy)

[Section 7 - User Interface Design](#_q2p4i71pnnm7)

[7.1 User Interface Design Overview](#_92yhlpkxkggg)

[7.2 User Interface Navigation Flow](#_v5yi3wwpwf5c)

[7.3 Use Cases / User Function Description](#_547gxdax3hhf)

[Section 8 - Other Interfaces](#_t2xxswkwylyn)

[8.1 Interface X](#_3cjmeucsrpv6)

[Section 9 - Extra Design Features / Outstanding Issues](#_fv06k0tsz5p6)

[Section 10 – References](#_zigwkt1lv1o6)

[Section 11 – Glossary](#_k7zsnsc4e2v7)

# Section 2 - Overview

## 2.1 Purpose

The primary purposes of the Elevate OLS-TT are twofold. First, to automate using LabVIEW, making the current manual ATP testing method more efficient; and second, to collect and record more precise data during ATP testing.

## 2.2 Scope

The Elevate OLS-TT roadmap consists of modules of functionality to be phased in as they are completed as follows:

1. Single Sensor test with voltage divider type measurement circuit. The intent of this first implementation is to prove the concept and produce valid data for engineering evaluation.
2. Multiple sensor test with aircraft compliant type measurement circuit, and pump speed control on a custom interface board. The intent of this implementation is to be ATP compliant and approved by the customer.
3. Variations that aid in the production processes, for example, pre-potting verification of switch installation and adjustment of the PCB assembly position.
4. Modifications to the basic test protocol and equipment will be migrated to other level sensor products.

In this current writing, only the first phase with be addressed.

## 2.3 Requirements

System requirements are specified in the Requirements Document *Elevate OLS ATP Requirements 2-7-24 (003).*

# Section 3 - System Architecture

Describe/include a figure of the overall system architecture (and where this module fits in)

A diagram of a computer

Description automatically generated

# Section 4 - Data Dictionary

Brief description of each element in this module or a link to an actual data dictionary

(template of a database table description)

|  |
| --- |
| **Table** |

|  |  |  |
| --- | --- | --- |
| **Field** | **Notes** | **Type** |
| ID | Unique Identifier from TABLE\_SEQ | DECIMAL |
| NAME | The Name in Object.Name() | VARCHAR |
| VALUE | The Value output from somewhere | VARCHAR |

# Section 5 - Software Domain Design

## 5.1 Software Application Domain Chart

Describe / chart each major software application domain and the relationships between objects (UML, etc)

## 5.2 Software Application Domain

A Comprehensive high level description of each domain (package/object wherever it is better to start) within the scope of this module (or within the greater scope of the project if applicable)

### 5.2.1 Domain X

A high level description of the family of components within this domain and their relationship. Include database domain, stored procedures, triggers, packages, objects, functions, etc.

#### 5.2.1.1 Component Y of Domain X

Define Component Y, describe data flow/control at component level

##### 5.2.1.1.1 Task Z of Component Y1 of Domain X

Define Task Z, describe data flow/control at task level

# Hardware Design

## DataQ 2108

### Sample Rate

The DI-2108 maintains an internal scan list of as many as eleven different items at once. The maximum sample throughput rate is 220 kHz, 20 kHz per scan list element with all positions enabled. Analog channels support a maximum sample throughput rate of 160 kHz, one analog channel at 160 kHz, two analog channels 80 kHz per channel, etc. The scan list can be populated with any combination of analog input channels, digital input ports, the rate channel, and the counter channel.

## Pump Controller-7

# – Data Design

Describe the data contained in databases and other shared structures between domains or within the scope of the overall project architecture

## 6.1 Persistent/Static Data

Describe/illustrate the logical data model or entity relationship diagrams for the persistent data (or static data if static)

### 6.1.1 Dataset

Describe persisted object/dataset and its relationships to other entities/datasets

### 6.1.2 Static Data

Describe static data

### 6.1.3 Persisted data

Describe persisted data

## 6.2 Transient/Dynamic Data

Describe any transient data, include any necessary subsections

## 6.3 External Interface Data

Any external interfaces’ data goes here (this is for the data, section 8 is for the interface itself)

## 6.4 Transformation of Data

Describe any data transformation that goes on between design elements

# Section 7 - User Interface Design

## 7.1 User Interface Design Overview

Pictures, high level requirements, mockups, etc.

## 7.2 User Interface Navigation Flow

Diagram the flow from one screen to the next

## 7.3 Use Cases / User Function Description

Describe screen usage / function using use cases, or on a per function basis

# Section 8 - Other Interfaces

Identify any external interfaces used in the execution of this module, include technology and other pertinent data

## 8.1 Interface X

Describe interactions, protocols, message formats, failure conditions, handshaking, etc

# Section 9 - Extra Design Features / Outstanding Issues

Does not fit anywhere else above, but should be mentioned -- goes here

# Section 10 – References

Any documents which would be useful to understand this design document or which were used in drawing up this design.

# Section 11 – Glossary

Glossary of terms / acronyms

# Section 10 – References

# Configuration Management

Configuration Management practice of tracking and controlling changes to a system.

## LabVIEW Development System

## Software Versioning

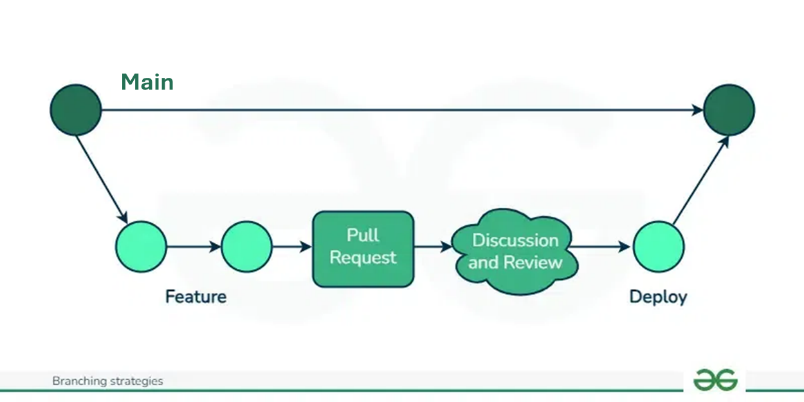
Version indicator consisting of two values the first is letter representing the Major revision and a number representing a minor revision.

* Major revision indicates the following applicability of the code:
  + Where E (Engineering) versions are for development evaluation only.
  + Where Q (Quality) versions are for interdepartmental evaluation but not for production, often referred to as Beta software.
  + Where letters A-D, F-P represent code has been qualified and released for production, the first release starts with A then incremented with a major release defined as not 100% backwards compatible (form, fit and function) with the prior release.
* Minor revision are two numbers, incremented with any change that remains 100% backwards compatible, (not materially affecting form, fit or function.)

## Git Branching Strategy

A branching strategy is a strategy that software development teams adopt for writing, merging and deploying code with the help of a version control system like Git. Since this project is likely to have a single developer the simpler, streamlined branching Github Flow strategy shall be used.

* **Main Branch** The primary branch of a Git repository where the most recent stable, production-ready code resides.
* **Feature Branch** is a development branch, created to work on a specific feature or task isolated from the main branch. Feature branch code distributed for deployment versioned as the Next release version prefixed with a Q. For example, if the Main Brance is version A02, then the Feature branch implementing a minor change shall be versioned QA03.
* **Merge:**The process of combining changes from one branch into another. In the case of using LabView, the simplest merge would be to simply replace the Main branch with the Feature branch when released.

****