Feature Template

*Revision: 0.0­*

*Document Release Date: 1/01/2017*

*Feature Integration Date: 1/01/2017*

*Document Owner: Joseph Tarango*

*Architecture Owner: Joseph Tarango*

*Development Owner: Joseph Tarango*

*Validation Owner: Joseph Tarango*

**Intel Confidential or Secret**

Contents

[Revision History 4](#_Toc468097704)

[1. Abstract 5](#_Toc468097705)

[2. Keywords 6](#_Toc468097706)

[3. Introduction 7](#_Toc468097707)

[3.1 Abbreviations, Acronyms, and Terminology 7](#_Toc468097708)

[3.2 Purpose, Scope, and Feature Request 7](#_Toc468097709)

[3.3 References and Related Documents 7](#_Toc468097710)

[3.4 Background and Related Work 7](#_Toc468097711)

[4. Requirements 8](#_Toc468097712)

[4.1 Problem Statement 8](#_Toc468097713)

[4.2 Competitive Analysis 8](#_Toc468097714)

[4.3 Entry Criteria 8](#_Toc468097715)

[4.4 Solution Space 8](#_Toc468097716)

[5. Architecture Design 9](#_Toc468097717)

[5.1 Algorithm Summary 9](#_Toc468097718)

[5.2 Explanations and Overview 9](#_Toc468097719)

[5.3 Structural Components 9](#_Toc468097720)

[5.4 Expected Behavior 9](#_Toc468097721)

[5.5 Special Conditions 9](#_Toc468097722)

[6. Design Construction 10](#_Toc468097723)

[6.1 Component Detail 10](#_Toc468097724)

[6.2 Data Representation 10](#_Toc468097725)

[6.3 Interface 10](#_Toc468097726)

[6.4 Extendibility 10](#_Toc468097727)

[6.5 Sequence Diagram 10](#_Toc468097728)

[6.6 Control and Data Flow 11](#_Toc468097729)

[6.7 Feature Dependencies and Interactions 11](#_Toc468097730)

[6.8 Performance Bottlenecks 11](#_Toc468097731)

[7. Deliverables 12](#_Toc468097732)

[7.1 Roles and Responsibilities 12](#_Toc468097733)

[7.2 Challenges 12](#_Toc468097734)

[7.3 Targets, Timeline, and Landing Zones 12](#_Toc468097735)

[8. Experimental Results 13](#_Toc468097736)

[8.1 System Setup 13](#_Toc468097737)

[8.2 Data and Results 13](#_Toc468097738)

[8.3 Analysis and Measured Behavior 13](#_Toc468097739)

[8.4 Conclusion 13](#_Toc468097740)

[9. Validation 14](#_Toc468097741)

[9.1 Tool Support 14](#_Toc468097742)

[9.2 Exit Criteria 14](#_Toc468097743)

[9.3 Validated Performance 14](#_Toc468097744)

[9.4 Competitive and Product Comparison 14](#_Toc468097745)

[10. Challenges Discovered and Forward Looking 15](#_Toc468097746)

[11. Acknowledgements, Contributors, and Reviewers 16](#_Toc468097747)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Author | Description | Date |
| 0.01 | Joseph Tarango | Format template and style for initial sample capture | 01/01/2017 |
|  |  |  |  |
|  |  |  |  |

# Abstract

Abstracts have several key components including: motivation, problem statement, approach, results, and conclusions. The purpose is to give a quick, clear, and concise overview of the subject area. Motivation is typically why do we care about the problem, problem statement is what we are trying to solve, approach is the methodology to go about solving the problem, results are the condensed answer to the problem, and the conclusion are the implications to the answer received.

# Keywords

Documentation, Expectations, and Formatting.

# Introduction

## Abbreviations, Acronyms, and Terminology

* FAS– Firmware Architecture Specification
* ISE – Internal Solid state drive Engineering
* SW – Software

## Purpose, Scope, and Feature Request

Describe the algorithm as defined for the program. A high-level overview of the core algorithm is provided while detailed design or implementation information for how these algorithms are embodied in the firmware codebase is left to the firmware design and FAS documents.

## References and Related Documents

|  |  |  |
| --- | --- | --- |
| Title | Description | Author |
| [Taylorsville SSD NAND Policy Rules](file://amr/ec/proj/fm/NSG/arch/Shares/Staff/CoreArch/Gen3ArchBoard/Doc) | NAND policy document for the Taylorsville client SSD program | Xin Guo |
| [Gen3 Core Algorithm Overview: Write Streams](file://amr/ec/proj/fm/NSG/arch/Shares/Staff/CoreArch/Gen3ArchBoard/Doc) | Algorithm overview description of the write streams and temperature separation | Knut Grimsrud |
| [Gen3 Core Algorithm Overview: Defrag Merit](file://amr/ec/proj/fm/NSG/arch/Shares/Staff/CoreArch/Gen3ArchBoard/Doc) | Overview of the merit equation used for selecting the target bands to be defragged and recycled | Neal Mielke |
| [Gen3 Firmware Power Loss Recovery Algorithm](file://amr/ec/proj/fm/NSG/SE/Shares/Taylorsville/Firmware/Docs/Design/Core/Context%20Lib) | Power loss replay specification | Suhas Nayak |

## Background and Related Work

The purpose of the template is to provide a framework so ISE can develop and execute more effectively. In the past there had been no standard in firmware development due to small team size. In order to scale, adequate documentation, design, and methodologies are required. Typically, related work is a literature review to highlight work done by others which ties to the active work. The review maybe the same work the current design is based on or attempts to solve the same problem space. In the related work process, gather the most significant related work internal or external to ISE and write about the details and explain the contributions and relation to current development.

# Requirements

## Problem Statement

Brief description of the issue to be addressed by answering the following questions:

* What is the problem?
* When does the issue manifest? Describe the specific issues of the problem.
* Where is the issue occurring?
* Who does the problem affect?
* What are the bounds of the problem?
* What is the impact of the issue causing? Why are we fixing it?
* What does the world look like if the problem is solved?

We want all of our software releases to be released to customer seamlessly, without defects, where everyone aware and informed of the outcomes and status.

## Competitive Analysis

The marketing and strategic assessment of the strengths, weaknesses, and the potential of competitors. The analysis provides both an offensive and defensive identification of the opportunities, possibilities, and jeopardies. The analysis is essential to being at the cutting edge of innovation such that we develop a proactive and the systematic approach to being ahead of the competition.

## Entry Criteria

A description of the system in place and to be developed. The criteria is the functional and non-functional requirements and use cases of the interaction the feature is to provide. We should provide the data item description defining the intended use with the primary objective. These all include a description of the deliverable, requirements to meet the contract specification, and product family.

## Solution Space

The solution space is the search space of all possible choices of a problem in order to satisfy the constraints and potentiality of each approach. The results should take into account the theoretical maximum, realist project constraints, feasibility, complexity, and timeline to approaches. From the approaches, a candidate solutions of the member set should be laid out with benefits and challenges. The explanations should be given in a breakdown of the motivation and feasibility.

# Architecture Design

## Algorithm Summary

**Inputs**

Sequence of array elements

**Outputs**

Single local maximum

**Algorithm**

Linear search (beginning, end)

If element i of known is greater current found element then replace

**Runtime**

Big-O(N) – Upper bound

Big-Omega(1) – Lower bound

Big-Theta(N) – Average Case

## Explanations and Overview

When defining algorithms, a paired explanations and overview should be elaborated to service understanding and clarity of the inputs, outputs, detailed inner workings, and runtime analysis. These details are important to understand the meeting of contractual agreements, specifications, and the general solvable case.

## Structural Components

Structural components is a branch of engineering where the focus is the separations of functionality into modules given a system. The basis is to reuse previous components to define and implement new functionality with interdependent subcomponents. The approach allows for short term and long term designs building upon the previous generation. The system modeling is based on realistic use cases and attempts to re-use as many components as possible with extensibility.

## Expected Behavior

In validating components the expected behavior is valuable based on the outputs of the system for key test cases. Based on the selected inputs the expected behavior acts as a black box verification technique for temporal properties. The modeling of the behavior allows for a methodical way of eliciting the requirements and putting them together for a specific design.

## Special Conditions

When designing a component, we target for the average case or general case accelerating those components. However, in order to be comprehensive and make a feature productizable we must cover all use cases including the diabolical situations. Therefore, in that respect the special conditions can be treated as slow path interactions where we gracefully handle the case at hand to ensure the overall product is functional.

# Design Construction

## Component Detail

A component itself is a domain solution to a known problem; whereas component level design is a modular and replaceable subsystem within an implementation. Each component encapsulates control and data manipulation representation based on architectural design specifications.

## Data Representation

Based data exchange models, each component will use and reuse best known data transfer formats to meet transactional based throughput models. For example, each element can be processed individually in a single burst transition or a vector burst where a set of data will all be manipulated at once to meet customer performance requirements. When no standard exists, a composition should be chosen to either transfer between intra-modules in an expanded format or compressed for external component boundaries. For continuous and power loss recovery, structured storage of critical data structures need to be considered to continue operations from given system interrupts.

## Interface

To standardize the exchange model, component-to-component transfer must reuse existing data structures in an efficient manner operating at a low memory foot print and fast data processing. Based on locality, each component should contain its own local variables that are not used in continuous data flow; however, are only necessary for the immediate computation. Each interface shall allow for data pass through or manipulation models. Data pass through models are simple due to moving the data to the correct component in control flow is based on decision graphs. Data manipulation models on the other hand, should be able to append and manipulate control structure to ensure the data is transferred to the correct end point.

## Extendibility

To ensure future proofing, data models should allow for the appending upon data structures in a minor revision and major revisions will change assumptions within the component level design. Each component interface will allow the passing of disjoint data structures such that backwards compatibility is maintained with minor version; however, major versions will require potentially new interfaces.

## Sequence Diagram

To maintain cohesion, each set of components needs to be grouped within the same utility patterns. The utility patterns are: functional, layering, temporal, procedural, and communication. Functional based cohesion is a direct one-to-one mapping of component manipulation. Layered is based on a set of components related to the same functional procedure accessing varying servicing components and to maintain adequate communication each operation on similar data subsets are to be grouped. Procedural and temporal cohesion is the ordering of groups in order to achieve a particular functionality such that all functional groups are able to use internal or external data communication methods. The sequence diagram demonstrates how components handshake operations between control and data driven events.

## Control and Data Flow

Based on the operations of the sequencing, each system can flow operations and/or manipulate data. For these two reasons, understanding the semantics of the intended sequence of events through control flow and the destinations of each data set through data flow is necessary. Understanding the control flow of a particular data set gives a good system understanding of events and operations; while the data flow analysis understands where data will propagate based on a given set of operations. Separating the two flow diagrams, clarify the behavioral or structural design; where the data flow will enhance local vs global data structures.

## Feature Dependencies and Interactions

For improved methodologies, components within the similar layering should be represented in a horizontal fashion; while interactions between components should be represented vertically. When taking into account, a time series each horizontal and vertical components can be used as a resource map then each control and data flow operation will be represented in a routing resource graph. The sequence of events, dependencies, and interactions between component layering and interfacing can be further segregated.

## Performance Bottlenecks

In tracing the resource constrained graphs, performance bottlenecks can be understood elaborating on design trade-offs for short and long term goals for generations or products.

# Deliverables

## Roles and Responsibilities

The list of people responsible for completion, validation, and verification of specifications related to feature.

* Requesters are the marketing and strategy members for the initial request
* Designers are the accumulation of architecture, system integrators, and engineers tasked with developing the feature.
* Constructors are the firmware system designers contributing and integrating methodologies to maintain extensibility to common models.
* Validators are case analysis evaluators for the focused and interface testing.
* Approvers are the product roadmap owners for the integration of product timelines and challenges encountered beyond technical methodologies.

## Challenges

Details of unforeseen issues which created additional methodologies, experimentation, change in expectations, etc.

## Targets, Timeline, and Landing Zones

One of the most critical components of design is the understanding of the sequential and parallelizable components based on product schedule. The roots of the idea of speeding up execution lies within Amdahl’s law. Amdahl’s law states the theoretical speedup of the latency of executing a specific task in a given a best known methodology is constrained by the bottlenecked or most sequential resources. In the instance of a specific task, we cannot exceed the time frame of the most constrained component; therefore, when planning a specific task we should create timelines pointing out the most sequential component. The timeline is a function based many unlisted factors and can be grossly simplified to the number of engineers resulting on the expected time frame

# Experimental Results

## System Setup

Explain system configuration such reproducing the expected results are easily obtained. System configurations can include hardware such as motherboard, processor, chipset, operating system, driver, firmware, preconditioning, and configuration files. Furthermore, if the requirement is designed for customer specific workloads, they should be listed and be given instructions. When new tests/scripts are designed then include a detailed description to execute the sequence of events.

## Data and Results

Data should be consolidated into a uniform format in raw data form and in an effective format depending on the evaluation. Each engineer is expected to have an intermediate ability to present data to peers; such as clearly labeled axis, titles, series, normalization when comparing deltas, and equations. When dealing with complex analysis break down data presented so the point is clear for each. Figure references within documentation and descriptions should be within the local presented area. When referencing these components within documentation the engineer should use the appropriate tools to link the paragraph. When in doubt, contact the technical lead for questions on how to compose and present data.

## Analysis and Measured Behavior

When the appropriate data is coalesced, the process of decomposing the area into smaller components to gain further incite, enlightenment, and understanding is analysis. The techniques used to decompose data into fundamental components is an entire field of study in mathematics. Each engineer should stride to make the digestion of concepts, functions, variables, algorithms, etc. as easy as possible so an engineer with a bachelor’s degree of study should be able to understand it. Exceptions due exist when complex numerical analysis in play so use your technical lead’s advice for such matters.

## Conclusion

One of the most critical components of any detailed analysis it the take away from the data. The conclusion is the restatement of the problem to the solution in a clear and concise summary. No new information is indicated from the construction of a conclusion; however, the data should inductively prove the solution indicated and leave a sense of closure for the feature.

# Validation

## Tool Support

Depending on the feature being developed, tool support could be critical to smooth continuous validation testing. Therefore in the architecture stage, the tool needed should be highlighted. Within ISE tool definition can be categorized such as internal test commands, NVMe command extension, validation tool support (I.E. swat, IOMeter, etc.).

## Exit Criteria

In preparation for the final testing of a specific features a methodology of black box, focus feature, and test harness based testing should be designed to reflect the state space/deterministic finite automata (DFA) model. The state space model is a physical representation of the expected linear system usually represented in an integer linear programming (ILP). The set of model equations will be based on the behavioral model and will be compared the firmware empirical model for accuracy. In cases where state space mathematics is unknown a DFA model, also known as a state machine (Mealy or Moore), can be represented such that we have a set of equations for each state and error condition coverage.

## Validated Performance

Upon completion of validated states and behaviors, the close of the expected time to completion in relation to customer specifications should be verified. The system level testing is to meet our expected current customer deliverables and active development.

## Competitive and Product Comparison

As a final conclusion to the product based feature, system integration should compare our expected behavior, performance, and error conditions to the current best known methods (BKMs). When BKMs are unknown, developers should attempt to communicate with customers such that the contract of the known behaviors are met or exceeded.

# Challenges Discovered and Forward Looking

Upon completion of any given feature, the discovered challenges, gaps, and future proofing development should be acknowledged. The documentation of the discovered work will aid in innovating and continuously improving our products for next generations. Furthermore, it give developers opportunities to contribute to new invention disclosure agreements such that the original authors have proof of contribution.

# Acknowledgements, Contributors, and Reviewers

The list of the developers and special thanks for going above and beyond expectations.