**Functional Specification**

**<Project Name>**

**Project Manager: <name>**

**<Team Members' Names, listed alphabetically>**

**Faculty Advisor: <name>**

**Texas State University**

**Ingram School of Engineering**

**SPONSOR Company Name**

**Street Address**

**City, State Zip Code**

**Date**



Remove this box and put an approved Sponsor logo in this space ONLY if your Sponsor approves doing so.

If they do not, center the UNM logo.

*To use this template:*

***Delete the contents of this page before submitting.***

The contribution of each team member must be clearly identified in this document. You may perform this identification in any manner **as long as it is clearly shown**.

**EXAMPLES**

**1.** *The 2.4 Error Handling section was written by John Doe.* <Executive Summary then follows this statement>

**2.** Identified by paragraph, table, etc.

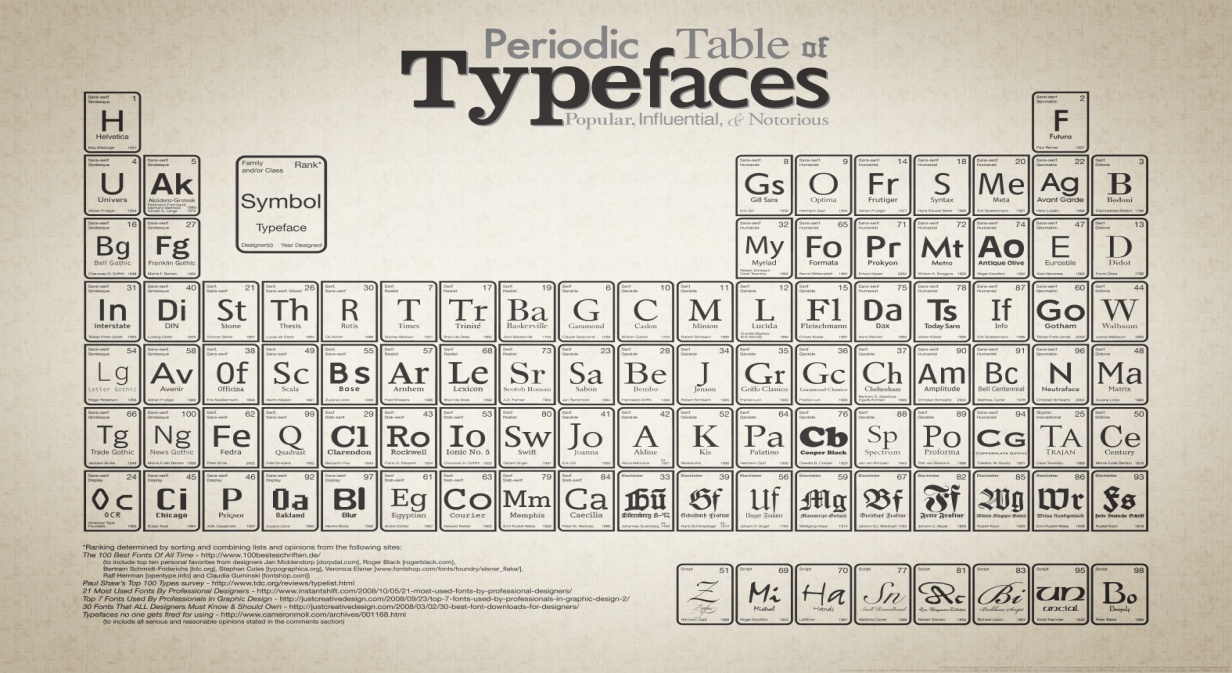
*Joaquin Doe:*

The Inverting Flurbitron is an extremely proliferous device in the magnetohydrodynamics area. While Flurbitrons have been available since the 1950's, it was only recently that Kygo & Selena shocked the world with the announcement of a reliable inverting Flurbitron. The applications are obvious.

*Julieta Doe:*

An alternative to solving the system problem with an Inverting Flurbitron is to use a Turbo Encabulator. The original machine had a base plate of prefabulated aluminite, surmounted by a malleable logarithmic casing in such a way that the two main spurving bearings were in a direct line with the pentametric fan. The latter consisted simply of six hydrocoptic marzlevanes, so fitted to the ambifacient lunar waneshaft that side fumbling was effectively prevented. The main winding was of the normal lotus-o-delta type placed in panendermic semi-bovoid slots in the stator, every seventh conductor being connected by a nonreversible tremie pipe to the differential girdlespring on the "up" end of the grammeters.

*Jose Doe:*



*To use this template:*

1. *Replace any red italicized text with your own text. You may add sections as needed for your particular projects, however, you must address each section* **even if NOT APPLICABLE***.*
2. *Enter the project name in the header.*
3. *If your document is very long, break each numbered chapter into its own document section, beginning it on a new page. This will make it easier to replace/update*
4. *Delete these instructions and any other italicized instructions.*

*Your documents must have an up-to-date revision block. The revision history block lists the changes that have been made (major revisions) to the document as the project progresses. Spelling corrections and other minor adjustments do not need to be listed. Entries should correspond to review cycles where the entire team reviews the document, not each individual edit. Add versions to the block as necessary; there may be more than 3. For many Senior Design documents just 1.0 will be fine. 1.0 is the version first signed off and submitted for grading.*

*Here are some possible revision numbers:*

*0.1 – Most sections identified, authors named*

*0.5 – Preliminary content for most sections*

*0.9 – Complete, number of changes diminishing*

*1.0 – Approval candidate, hopefully signed off*

*1.1 – Minor changes after sign off*

*2.0 – Major update, perhaps next gen product*

*Revision blocks are for you, the reviewers, and your customers. Documented changes allow everyone to understand latest information without rereading entire document.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| **Version** | **Date** | **Description** | **Author** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*Once you have completed the required information for each section, you must update the table of contents. It is linked to the section headers so DO NOT DO THIS MANUALLY. Right click anywhere on the table of contents > Left click “Update Field” > Left click “okay”. If you made your own document section, make sure it is reflected here. You may need to insert a Field.*

Table of Contents

[1 Introduction 4](#_Toc506200490)

[1.1 Summary 4](#_Toc506200491)

[1.2 Customer (or Sponsor) Requirements 4](#_Toc506200492)

[1.3 Existing System 5](#_Toc506200493)

[1.4 Terminology 5](#_Toc506200494)

[2 Functional Description 5](#_Toc506200495)

[2.1 User Attributes and Use Cases 6](#_Toc506200496)

[Who will be using this product? What are the skills / demographics? Will we provide everyone with the same functions? 6](#_Toc506200497)

[2.2 Administration Functions 7](#_Toc506200498)

[2.3 Error Handling 7](#_Toc506200499)

[2.4 Safety and Security 8](#_Toc506200500)

[2.5 Help and User Documentation 8](#_Toc506200501)

[2.6 Interfaces 9](#_Toc506200502)

[2.6.1 User 9](#_Toc506200503)

[2.6.2 Software 9](#_Toc506200504)

[2.6.3 Hardware 9](#_Toc506200505)

[2.6.4 Mechanical 9](#_Toc506200506)

[2.7 Boundary Conditions and Constraints 9](#_Toc506200507)

[2.8 Performance 11](#_Toc506200508)

[*Here’s an example on this page and the next.* 11](#_Toc506200509)

[2.9 Software Platforms 13](#_Toc506200510)

[2.10 Service, Support, & Maintenance 13](#_Toc506200511)

[2.11 Expandability or Customization 13](#_Toc506200512)

[3 Project Alignment Matrix 14](#_Toc506200513)

[4 References 15](#_Toc506200514)

[5 Approvals 15](#_Toc506200515)

*Introduction to the Functional Specification*

*In general terms, the functional specification states* ***what*** *the proposed system is to do, not* ***how*** *it’s to be done. However, in writing it, some consideration of design issues must take place, to ensure a realistic system is specified.*

***The functional specification should be clear, consistent, precise and unambiguous.***

*It is important that there is a draft functional specification before the design stage on any project is started and that the functional specification is agreed upon. There* ***must be a deadline*** *on the project plan for the issue of the functional specification. The functional specification* ***must be kept up-to-date****, as this is the communication with the world outside the development staff.*

*Use diagrams where appropriate.* ***I usually reject specs without diagrams.*** *Do not be afraid of examples!*

*Examples and diagrams are often better at communicating a complex idea than a lot of abstract or high level language. If a student not on your team reads a section and doesn’t understand it, you may consider including an example or diagram. Choose interesting examples to keep your reader’s interest!*

*Use consistent language and formatting in your diagrams. Although you may know that part A and part B are the same, your reader may not. Color scheme, font, and design should be kept consistent between diagrams. It is nice to have a visually appealing document, and a kaleidoscope looks terrible.*

# **Introduction**

*An introductory sentence or two about the project as this may be the first exposure to the project for some readers. Work done on the SOW will help you here!* ***START RIGHT AWAY AND STATE WHAT YOUR PROJECT IS GOING TO DO. SHOW A BLOCK DIAGRAM OF HOW IT FITS IN TO THE BIGGER PICTURE.***

## Summary

*A few sentences summarising the project: what it is, who it is for (customer or internal), is it a made-to-order project, a product, a demo. Should expand a bit beyond just the sentence or two in the paragraph above.*

## Customer (or Sponsor) Requirements

*This section should state the requirements the functional specification is attempting to fulfil. This may be an understanding of a customer’s requirement or a statement given as an internal starting point, e.g. produce a comprehensive mail tool in minimum time. Normally requirements are by their nature unstructured with high and low level statements intermingled. This section should refer to a separate requirements document if it exists. If there is anything else clarifying the requirement - such as the sponsor’s* **New Project Definition** *- these should also be referred to and probably a copy put into an Appendix.*

## Existing System

*This section should include an explanation of the system we are replacing, even if it’s an old manual system. If no existing system, state this.*

*What problems does the current system have? Which of these problems do we solve or reduce?*

*What useful functions of the current system will we not provide (Constraints)?*

*Depending on the depth of analysis required, this section may also describe the root causes of each problem. “Root cause” analysis is a systematic way of uncovering the underlying cause of an identified problem:*

*“It’s amazing how much people do know about the problem behind the problem; it’s just that no-one – by which we usually mean management – had taken the time to ask them before. So, ask them and then ask them again.”  
Source: Managing Software Requirements: A Unified Approach by Dean Leffingwell, Don Widrig – Chapter 4, “The Five Steps in Problem Analysis”*

## Terminology

*This section should contain all words, phrases, or acronyms having a special meaning for this project. Use a clear and concise statement of their meaning. List examples if necessary. The table below is a suggested format with examples = you need to make your own entries. HINT: This is a table with no borders – it is a good way to control the placement of entries, pictures, etc in Word. To see gridlines, click on table, select table tools, layout, view gridlines.*

|  |  |
| --- | --- |
| *Term* | *Description* |
| *TSGC* | *The Texas Space Grant Consortium* |
| *IGBT* | *Insulated-Gate Bipolar Transistor* |

# **Functional Description**

*The rest of the document may be divided into individual sections or chapters depending on the size and complexity of the system.*

*Pay attention to the flow of the document. Avoid forward references (such as, Refer to Page 565) and consider re-ordering of the document in such circumstances. If you’re having difficulty re-ordering, consider including the word or phrase in the terminology section.*

*The functions provided to fulfil the requirements must be structured.*

*All statements as to functionality should be written clearly using consistent terminology. Clearly describe the input, environment, and the result of the product so that someone else would be able to verify your results. This person should not have to guess or interpret any part of the description.*

*If not all functions are required for project success, differentiate between the ones that are mandatory and additional ones that would be nice to have.*

*Related functions may be logically grouped to make the document more readable.*

*Number each statement of a function.*

*All the following headings must be included somewhere in the document, not necessarily in the order given here.*

*If it is not relevant or we are not addressing it for this system, then say why.*

*It may make sense to group all of the irrelevant headings at the end.*

## User Attributes and Use Cases

*Who will be using this product? What are the skills / demographics? Will we provide everyone with the same functions?*

*Groups of people the product is targeted at are normally identified as use case roles (i.e. actors), and the functions assigned to each role as individual use cases. Where this information does not fit into the use case model, it should be captured in the main functional specification instead.*

*A use case is a sequence of interactions between actors in a system to achieve a goal.*

*Use cases should be listed using a unique and descriptive name. Depending on the size of the system being modelled, you might also need to include the package name.*

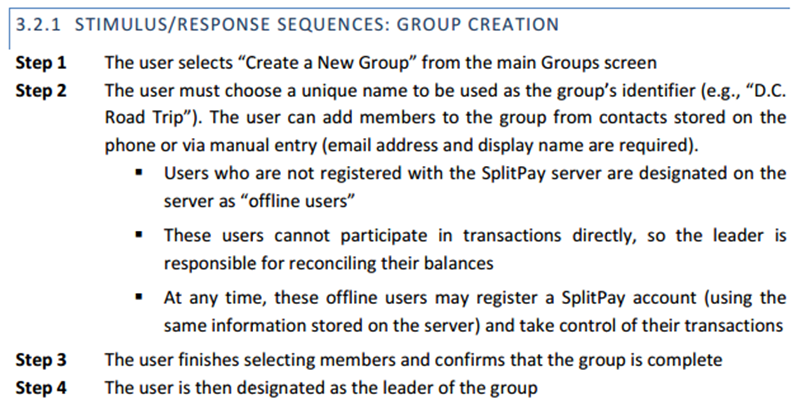
*If you have a lot of use cases, maintain them in a separate document and include it in the References section.*

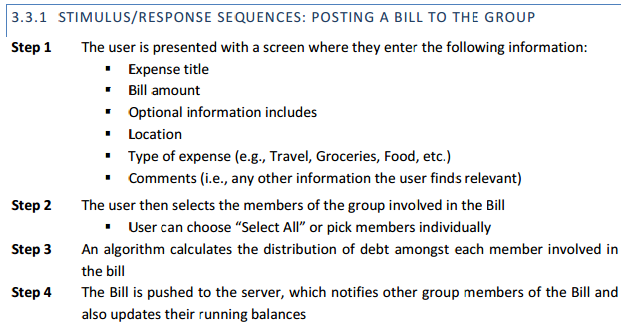
*The functional spec and uses cases feed into each other. Your functions can be sequenced into use cases and use cases might require additional functions.* ***BEWARE FEATURE CREEP.*** *Make sure that each use case is related to solving the sponsor’s need. Chances are, a manufacturing robot doesn’t need a Twitter account.*

*Avoid needless repetition. The amount of detail in the rest of the functional specification will depend on the number of use cases that have been written.*

*Functional requirements and boundary conditions belong in the Functional Spec. Limit any text in the Use Cases to usage scenario descriptions.*

*You may need to have sections which are sequential in nature, an example being:*

**



## Administration Functions

*Do all users of the system have equal access or does the system have administrators?*

*How will the system be administered?*

*Are there separate functions for an administrator?*

*What security is built in to stop others using administrative functions?*

*The administrator functions will be similar to the use cases but typically start with enter password…*

## Error Handling

*Identify the possible errors and how they should be handled. Every system needs this.*

*If there are different error types, group them logically and identify the reason for their classification.*

*Some common cases to consider:*

*No Network Connection*

*Battery Low*

*Storage Media Not Found*

*Storage Media Full*

*Sensor Readings Out of Bounds*

*The team should brainstorm and come up with a good list of the potential sources of errors and how they should be handled.*

## Safety and Security

*Safety and security considerations are an important part of any project. This section should detail possibilities of abuse of the system.*

*Do users have access to stored data? Can they modify the software? Can they change the configuration of the system? These functions are typically limited to administrators, if so how will access be granted.*

*Is the data sensitive or valuable? Should it be encrypted? Secured with a mechanical lock?*

*Are users able to interact with potentially dangerous components directly?*

*Along with error handling, the specification has to handle “the negative path”. There is no point in having a system that does lots of good things if it also does lots of bad things.*

*How will you limit things from going terribly wrong?*

*Example – if a system has video or audio recording it should have a clear and bright indicator LED when active.*

*Not all projects have significant safety and security considerations. If not – clearly state this and provide a short explanation of why not.*

## Help and User Documentation

*What type of help is to be provided, if any? Will you include user documentation as part of the system e.g. help menus or write a user guide?*

*Will you provide a reference guide with setup and operating instructions for use by the sponsor or future teams?*

*Do you want to answer project phone calls or emails after graduation?*

## Interfaces

*Four different interfaces are addressed below: User (to system), Software (within system, or to another system), Hardware (within system, or to another system), and Mechanical (within system, or to another system).*

### User

*This could be a chapter in its own right if it is a full definition. If it is deferred to the design specification stage, this should be stated. How will the user interface with the system? Touchpad? Buttons? Keyboard & Mouse?*

*Some things to consider:*

*How will the user know the system is on?*

*If battery powered, does it indicate charging, full, low?*

*Recall section on safety and security – are there appropriate indicators when recording?*

### Software

*We may be interfacing to existing software. This should be stated, e.g. toolkits, back ends of existing packages. State versions. Do interface documents exist?*

### Hardware

*We may be interfacing to existing hardware. This should be stated, e.g. toolkits, back ends of existing packages. State versions. Do interface documents exist?*

### Mechanical

*Describe the mechanical interfaces, versions, constraints, and specification documents. This is mostly intended for Interdisciplinary Projects, but may also apply to projects with mechanical components.*

## Boundary Conditions and Constraints

*Boundary Conditions represent the limits to which your product will be guaranteed to perform and serve as the basis for pass/fail determination during the testing phase. Constraints are fundamental limitations which cannot be exceeded. For instance, your system may need to work running on battery power from 2.5V to 3.3V. This is a boundary condition. If the maximum voltage of the part being powered is 5V and it will be permanently damaged if exceeded, this is a constraint.*

*Boundary conditions will always be less than or equal to the corresponding constraint. They can be used to limit the amount of testing required. If the product measures liquid water temperature under normal conditions, you may want to set the boundary conditions at 5 degrees Celsius minimum to 40 degrees Celsius maximum even if you temperature sensor has a much wider range.*

*If there is something limiting your project that you haven’t discussed elsewhere, include it here.*

*Some constraints might be economic, political, technical, system, environmental, or even scheduling constraints.*

*All projects have cost and schedule constraints. Some are forced to use specific hardware or software. These are constraints.*

*NOTE: This section will be scrutinized by ABET so pay some diligence here!! The information here should also be used to complete the constraint matrix near the end of the document.*

*You have to transmit wifi 600' to the office - that's a boundary condition. It can transmit 1200' maximum - that's a constraint.*

*You are sensing heart rate. You've set the alert limits at below 50 bpm and above 155 bpm - those are boundary conditions. The unit can measure from 1 bpm to 400bpm - that's its constraint.*

*You have a limited budget for your project so you can't afford a supercomputer to do the processing - that's a constraint. Because of this you can only process 2 frames per minute - constraint.*

***Confused? Need help?***

***1) Go to D2 Team***

***2) Go to D2 PM***

***3) Ask Faculty Sponsor***

## Performance

*This section should include a table of all performance parameters such as Capacity, Response time, accuracy, storage… to name a few. It will depend upon your specific project. The items will also depend upon the type of element you are specifying, i.e., code vs. hardware. Differentiate between the ones that are mandatory and additional ones that would be nice to have.*

*Here’s an example on this page and the next.*

*Place all of your performance categories and numbers in this section.* ***You’ll have more than just Capacity and Response times!***

*It will be helpful to use landscape mode!*

*Using your block diagram go through all of your inputs and outputs and figure out min*

*and max ranges. Quantitative values should be used!*

*Electrical Signals will have min voltage, max voltage, potentially tolerance and current*

*Battery life will have a minimum acceptable and potentially typical operating time*

*You should not specify a maximum battery life – it’s meaningless and hard to test*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Example Hardware Performance Parameters*** | | | | | |
| ***Parameter*** | ***Test Conditions*** | ***Min*** | ***Max*** | ***Units*** | ***How Tested*** |
| *Power Supply Rejection Ratio* | *Vs = ± 150V* | *70* |  | *dB* | *Measure output ripple with oscilloscope while injecting AC ripple into supply* |
| *Power Supply Voltage* | *Vin=0V* | *± 20* | *± 170* |  | *Measure Q-Point with voltmeter over supply range* |
| *Output Short Circuit Current* | *Vs = ± 150V* |  | *40* | *mA* | *Short output to ground using a 1Ω resistor, measure voltage to infer current* |
| *Bandwidth* | *Vs = ± 150V*  *Rs = 10Ω* | *1* |  | *MHz* | *Sweep input frequency to create magnitude vs. frequency plot* |
| *Linearity* | *Vs = ± 150V* | *3* | *1* | *%* | *Measure input vs. output voltage with voltmeter over range of input* |

|  |  |  |
| --- | --- | --- |
| ***Example Software Performance Parameters*** | | |
| ***Function*** | ***Description*** | ***How Tested*** |
| *New releases* | *When a new/updated version or release of the software is released, the user should be notified.* | *Construct updated version and verify that notification correctly occurs.* |
| *User interface* | *Loads in 5 seconds or less* | *Use a timer to measure load time* |
| *Memory usage* | *The amount of Operating System memory occupied by the application. Target is 10MB, limit is 20MB.* | *Observations done from the performance log during testing.* |
| *System reliability* | *The reliability that the system gives the right result.* | *Measurements obtained from 100 different usage instances during testing.* |
| *Title bars* | *Accurate and consistent titles will be displayed in all windows, frames, and dialogs* | *Each instance checked for accuracy and consistency* |
| *Dialogs* | *The user will be able to cancel all dialogs by hitting the [Esc] key* | *All dialogs tested by [Esc]* |
| *Access control* | *(a) If enabled, all users will be limited by their group assignments (b) If disabled, all users have free access to system resources* | *Check access with control both enabled and disabled* |
| *Platforms* | *The system will run on the following computer platforms: (a) Windows 10 (b) Windows 7* | *Test all functions listed above on all platforms* |

*You* ***WILL*** *have to test each of these in the future.*

## Software Platforms

*List which software platforms you will be supporting. Name a reference platform or platforms plus appropriate operating system versions.*

## Service, Support, & Maintenance

*Are any functions to be included to make maintenance and support easier, e.g. internal monitoring of traffic flows, diagnosis mode, etc. This typically applies to pushing down software updates.*

*Is the system designed to be serviceable with elements such as easily accessible battery, and modular components or is it sealed for durability reasons?*

## Expandability or Customization

*State the likely expansion requirements, such as new users, markets, platforms, or features. Refer to any items that have already been considered as well as any additional items.*

*Can the user to customize the system? If so, what are we going to provide? Source code? Configuration menus?*

# **Project Alignment Matrix**

*Ask your Faculty Advisor to assist you in the completion of these tables below.*

***Do NOT add or delete Course No’s (Table 1) or Constraint Types (Table 2).***

*Fill in the right columns. If an entry doesn’t apply, state why it does not apply.*

Outside Advisors (if any) and affiliations:

**TABLE 1: Knowledge Alignment Matrix**

|  |  |  |
| --- | --- | --- |
| **Course No.** | **Core knowledge** | **Specific knowledge incorporated by team** |
| EE 3350 (Electronics I) | Design and analysis of active devices and equivalent circuits |  |
| EE 3370 (Signals and Systems) | Frequency domain representation of signals and frequency response, transfer functions |  |
| EE 3420 (Microprocessors) | Principles of operation and applications of microprocessors |  |
| EE 4352 (Introduction to VLSI Design) | Analysis and design of CMOS integrated circuits |  |
| EE 4370 (Communications Systems) | Transmission of signals through linear systems, analog and digital modulation, and noise |  |

**TABLE 2: Constraint Alignment Matrix (and applicable standards)**

ABET Criterion 3 (c): “an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.”

|  |  |
| --- | --- |
| **Constraint Type** | **Specific Project Constraint** |
| Economic |  |
| Environmental |  |
| Health and safety |  |
| Social/Ethical |  |
| Applicable Standards |  |

Need help with this Matrix?

1. Go to D2 Team

2. Go to D2 PM

3. Go to your Faculty Advisor

# **References**

*List any document references with numbers. Refer to them as* ref[n] *in the rest of the document. Remembering to include issue numbers and/or dates so that the actual version is identified.*

*If your software has documentation, list it here.*

*If your component has a spec sheet, list it here.*

*If you are interfacing with hardware or software and that hardware or software has documentation, list it here.*

*If there is a survey, study, or article that you use in your Functional Spec for support, list it here.*

*When including web links, link directly to the document if possible:*

*Use this:*

*[1] Maxim LM75* *Digital Temperature Sensor and Thermal Watchdog with 2-Wire Interface Datasheet.* [*https://datasheets.maximintegrated.com/en/ds/LM75.pdf*](https://datasheets.maximintegrated.com/en/ds/LM75.pdf)

*Not this:*

*[1] LM75 datasheet at www.maximintegrated.com*

# **Approvals**

*The project Functional Specification is a document with which approval is granted or denied to move forward. Therefore, the document should receive approval or disapproval from its review board.*

The signatures of the people below indicate an understanding in the purpose and content of this document by those signing it. By signing this document you indicate that you approve of the proposed project outlined in this Functional Specification and that the next steps may be taken to proceed with the project.

|  |  |  |  |
| --- | --- | --- | --- |
| **Approver Name** | **Title** | **Signature** | **Date** |
|  | Project Manager |  |  |
|  | D2 Project Manager |  |  |
|  | Faculty Sponsor |  |  |
|  | Sponsor |  |  |
|  | Instructor |  |  |