[Pick the date]

|  |
| --- |
| Driver Interface V1.0 | Fredrick Omondi |



|  |  |
| --- | --- |
| Waterford Institute of Technology /Vector GB | Design Specification – Driver Interface |

Table of Contents

[Subsystem Requirements Summary 2](#_Toc341359646)

[System Context 2](#_Toc341359647)

[Software Architecture 2](#_Toc341359648)

[Node <name> 3](#_Toc341359649)

[Description 3](#_Toc341359650)

[Interfaces 4](#_Toc341359651)

[Interface with <name> 4](#_Toc341359652)

[Signals and Services 4](#_Toc341359653)

[Graphical User Interface 4](#_Toc341359654)

[Interface Design 4](#_Toc341359655)

[Interface with <name> 6](#_Toc341359656)

[Signals and Services 6](#_Toc341359657)

[Graphical User Interface 6](#_Toc341359658)

[Interface Design 6](#_Toc341359659)

[Node Local Data 7](#_Toc341359660)

[Function Design 8](#_Toc341359661)

[Top Level Flow Control 8](#_Toc341359662)

[Function <name> 8](#_Toc341359663)

[Function <name> 8](#_Toc341359664)

[Function <name> 8](#_Toc341359665)

[Test Case Design 9](#_Toc341359666)

[References 10](#_Toc341359667)

# Subsystem Requirements Summary

# System Context

This chapter describes the overall context (interfaces) to your system. It is essentially the top-level block of your subsystem from the architecture diagram. Include the complete diagram and give a general description of the interfaces to your block.

# Software Architecture

In this chapter you will take the input and output signals from the top-level block and decompose to the next level to include sensors, actuators and software function blocks. For this project it is probably sufficient to model software functions to the node level only as the implementations will not always follow a dataflow model as used by PREEvision. For example, an event-driven approach as used by CANoe CAPL code would be best described by a statechart and the iPad app best described using UML diagrams. Basically, each software block at this level is a node in your subsystem, running on CANoe for example and communicating with other nodes over the network. The internal functionality of each node is further decomposed later in the Function Design chapter.

# Node <name>

This section describes the complete design of one node in your subsystem (a software function from your software architecture diagram). If you have more than one communicating node in your software architecture then repeat this whole section for each node.

## Description

Briefly describe the function of this node.

## Interfaces

This section describes all of the interfaces between your node and the outside world and how you are going to implement those interfaces.

### Interface with <name>

A brief description of what this interface is used for and what it communicates with (e.g. hardware or other nodes).

#### Signals and Services

A description of each signal that passes across the interface (in the case of network interfaces) or else services accessed using function calls (for example to a DLL). For signals (sender-receiver interfaces), the name of the signal (as per the CANdb or FIBEX file), the data type and a usage description should be given.

For function calls (client-server interfaces), list the name of the service together with a high level description of each input and output parameter.

Tables would be a good layout option for this section.

#### Graphical User Interface

If your subsystem includes a GUI element (iPad app or touch screen display for example) describe in this section all of the screen layouts that will be used. Try to give actual prototype screen shots if possible.

Where user input is allowed/expected describe all of the allowable inputs (e.g. touch gestures, menu selections or data entry) and the application response to each input. For complex navigation between screens a statechart might be a useful description technique.

#### Interface Design

Include specific details of how you will access this interface from your code. For signal interfaces explain how you are going to access the network interface (e.g. CAN bus, MOST etc) to send and receive signals. Give the actual API calls for any DLL functions or object methods (e.g. Gadgeteer methods) that you will need.

For function calls to DLLs give the actual API calls including all parameters, return values and their descriptions.

### Interface with <name>

A brief description of what this interface is used for and what it communicates with (e.g. hardware or other nodes).

#### Signals and Services

A description of each signal that passes across the interface (in the case of network interfaces) or else services accessed using function calls (for example to a DLL). For signals (sender-receiver interfaces), the name of the signal (as per the CANdb or FIBEX file), the data type and a usage description should be given.

For function calls (client-server interfaces), list the name of the service together with a high level description of each input and output parameter.

Tables would be a good layout option for this section.

#### Graphical User Interface

If your subsystem includes a GUI element (iPad app or touch screen display for example) describe in this section all of the screen layouts that will be used. Try to give actual prototype screen shots if possible.

Where user input is allowed/expected describe all of the allowable inputs (e.g. touch gestures, menu selections or data entry) and the application response to each input. For complex navigation between screens a statechart might be a useful description technique.

#### Interface Design

Include specific details of how you will access this interface from your code. For signal interfaces explain how you are going to access the network interface (e.g. CAN bus, MOST etc) to send and receive signals. Give the actual API calls for any DLL functions or object methods (e.g. Gadgeteer methods) that you will need.

For function calls to DLLs give the actual API calls including all parameters, return values and their descriptions.

## Node Local Data

Describe the internal data which your node will need to maintain in order to carry out its functions. This could include things like temporary data storage, buffers, calculated data, user preferences etc. You could model this as a UML class diagram in Visual Paradigm to show types of data (classes), attributes stored and what has to be done with the data (methods). You don’t need to implement the data as objects but taking an object-oriented view may help clarify the design. If you are using an MVC design (e.g. iPad app) then this section will describe the Model data.

## Function Design

### Top Level Flow Control

Top level statechart or sequence diagram of the node’s main program and function calls.

### Function <name>

1. Purpose (brief description)
2. API signature details (input, output parameters and return values)
3. Logic description (outline code, sequence diagrams etc)

### Function <name>

1. Purpose (brief description)
2. API signature details (input, output parameters and return values)
3. Logic description (outline code, sequence diagrams etc)

### Function <name>

1. Purpose (brief description)
2. API signature details (input, output parameters and return values)
3. Logic description (outline code, sequence diagrams etc)

# Test Case Design

In this chapter describe the procedure by which you will test the functionality of your nodes and subsystem. Try to list the main test cases in general terms and describe any tools or techniques that will be needed to test your subsystem. For example, HIL systems, test case generators, simulators such as CANoe etc.

# References

List using Harvard notation any documents, datasheets etc. that provide useful background information.