Technical Manuel
Yavin IV Defence System

Team Dirac

October 31, 2014

Chapter 1

Introduction

1.1 Document Identification

This document describes the design of the Yavin IV Defence System. This document is prepared by Group DIRAC for assessment in MTRX3700 in 2014.

1.2 System Overview

¡A brief statement of the purpose of the system or subsystem to which this document applies¿ The Yavin IV Defence System is designed to provide accurate, low cost tracking for Death Stars and other similar objects.

1.3 Document Overview

¡A short "road map" of the document, to provide an orientation for the reader. Summarise the purpose and contents of this document.¿ This document describes the detailed technical specifications and functionality for the entire system. This includes the entire system design, implementation and usage.

1.4 Reference Documents

The present document is prepared on the basis of the following reference documents, and should be read in conjunction with them. ¡Insert relevant documents¿

1.4.1 Acronyms and Abbreviations

Acronym	Meaning
Thing	Meaning of Thing
Stuff	Meaning of Stuff

Chapter 2

System Description

This section is intended to give a general overview of the basis for the Yavin IV Defence System system design, of its division into hardware and software modules, and of its development and implementation.

2.1 Introduction

¡Give a technical description of the function of the whole system, in terms of its constituent parts, here termed modules. Generally, a module will have hardware and software parts.

2.2 Operational Scenarios

¡Describe how the system is to be used. There may be several different ways that it ca be used perhaps involving different users, or classes of user. Present case diagrams here if you are using them. Each operational scenario is a part through a use case diagram - a way of using the system, with different outcomes or methods of use. You should also consider the various failures that may occur, and the consequences of these failures. ¿

2.3 System Requirements

The operational scenarios considered place certain requirements on the whole Yavin IV Defence system, and on the modules that comprise it. ¡Statement of requirements that affect the system as a whole, and are not restricted to only a subset of its modules.¿

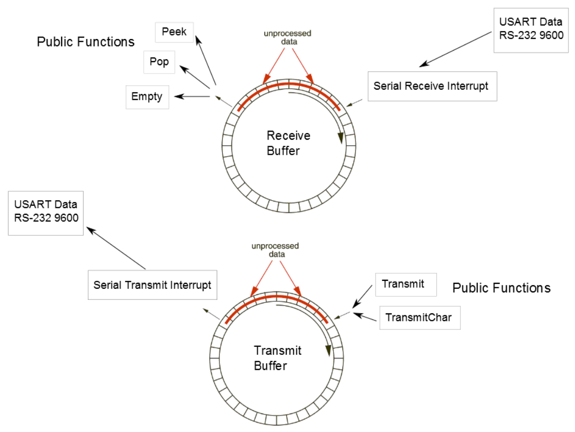

Figure 2.1: Shows Conceptual Diagram of Serial Module -¿ Received input stored by interrupts into circular buffer to await pop commands. Transmit data pushed onto buffer which is transmitted via interrupts

2.4 Module Design

¡Describe the breakdown of the design into functional modules. Each module probably contains both software and hardware. Then include a section like the following 2.5 for each module. Not all of the sub-headings may be relevant for each module.¿

2.5 Serial

2.5.1 Description

The serial module takes care of all communication (transmit and receive) over the serial UART (rs-232) port.

2.5.2 Functional Requirements

The serial module conforms to the following functional requirements:

- Fully interrupt driven transmit and receive
- Separate circular buffers to store data received and data to be transmitted
- Functionality to add data to the transmission buffer, read from received buffer, and check if anything has been received

2.5.3 Conceptual Design:

The serial module is INTERRUPT DRIVEN. This means that any background code can be running while the module is transmitting and/or receiving data over the serial line, and no serial data should ever be missed, overlooked or cut out.
The module contains two circular buffers: a transmit buffer and a receive buffer. These buffers are NOT accessible by the rest of the program. Rather, the module provides a public function transmit() which takes a string, and places it into the transmit buffer. Anything in this buffer is then transmitted character by character when the transmit ready interrupt fires.
Whenever a character is received over serial it is stored in the received buffer by an interrupt. Again this buffer is NOT accessible to the rest of the program. Rather it provides a number of functions to interact with it. The most commonly used of which is the readString() function, which returns everything in the buffer up to a carriage return (e.g. a line of input entered by the user).
This serial module also allows users the opportunity to remove or change data they have already transmitted. If a backspace is received, then instead of storing it in the receive buffer it will remove the last received character from the buffer if that character is not a Carriage Return, Newline, or Escape operator. This enables a much more more user friendly system as otherwise there would be no way to fix any syntax error without pressing enter, getting an error and starting again. Furthermore, without this feature, if a user did backspace and change an input it would result in completely unexpected behaviour.