CLI INFRASTRUCTURE FOR EMBEDDED DEVICES

High level requirements document

1. Introduction

This document outlines the high-level requirements for developing a Command Line Interface (CLI) infrastructure suitable for embedded systems. The CLI is designed to register and process various commands, including a fixed 'help' command as the first entry.

2. Objectives

- To create a flexible and efficient CLI system for embedded devices.
- To enable easy registration and processing of custom commands.
- To implement a consistent and user-friendly interface for command interaction.

3. System Overview

The CLI infrastructure will consist of:
- A command registration system.
- A command processing mechanism.
- A predefined 'help' command.
- User-defined commands.

4. Functional Requirements

4.1 Command Structure

Each command will be defined in a structure, including:
- Command name.
- Help text.
- Command interpreter function.
- Expected number of parameters.

4.2 Command Registration

The system will provide a function to register commands. Commands will be stored in an array or similar data structure. There will be a configurable limit to the number of commands that can be registered.

4.3 Command Processing

The system will parse input strings to identify and execute commands. On identification, the corresponding command interpreter will be executed. Appropriate feedback or output will be provided for each executed command.

4.4 'Help' Command

The 'help' command will always be the first command in the system. When invoked, it will list all registered commands and their descriptions. The 'help' command cannot be modified or unregistered.

4.5 User-Defined Commands

Users can define and register their own commands. Each command can have a unique set of parameters and functionality.

5. Technical Requirements

5.1 Language and Libraries

The system will be implemented in C. Standard libraries like <string.h>, <stdint.h>, and <stdio.h> may be used.

5.2 Memory Management

Efficient use of memory is crucial, considering the resource constraints of embedded systems.

5.3 Error Handling

The system will robustly handle errors such as unrecognized commands or exceeded command limits.

6. Testing and Validation

Each component (registration, processing, etc.) will undergo thorough testing. The complete system will be tested for cohesive functioning. Testing under various loads to ensure stability and reliability.

7. Documentation

Comprehensive documentation should be provided, covering usage, examples, and API details.

8. Future Scope

Expansion to include more complex command parsing. Potential integration with network modules for remote CLI access. Incorporation of encryption for password and critical information. Addition of a console interface with virtual terminal semantics.