Getting Started

- Directory Layout CORE8-16F Version

Directory Layout CORE8-16F Version

Directory Structure Documentation for CORE8-16F Version

Overview

This documentation provides an overview of the directory structure of the CORE8-16F microcontroller framework. The purpose of this guide is to help developers quickly understand the framework's layout, where to find specific components, and the function of each directory and file.

Top-Level Directory (`src/`)

- **`main.c`**: The main entry point of the program. This file contains the initial setup and execution flow for the framework.

Core Framework Directory (`core16F/`)

- **`core16F.h`**: The primary header file for the CORE8-16F framework, providing core definitions and function prototypes.
- **`core16F_const.h`**: Defines constants used throughout the core framework, ensuring consistent values are used across different modules.
- **`core16F_init.c`**: Initialization routines for the CORE8-16F, setting up essential configurations before main program execution.
- **`core_version.h`**: Tracks the version of the core framework, providing version information that can be used for compatibility checks.

Core System Subdirectory (`core16F_system/`)

- **`delays/`**: Provides delay routines with `delays.c` and `delays.h` for timing control in the program.
- **`events/`**: Manages system events, with `events.c` and `events.h` handling event triggering and processing.
- **`timer/`**: Includes files (`isr_core16F_system_timer.c`, `.h`) for handling system timer interrupts, crucial for timing operations and event scheduling.
- **`utils/`**: Utility functions (`utils.c`, `utils.h`) that provide common operations used throughout the core, like math functions or data conversions.

Core Interface Documentation

The core interface provides essential functions and definitions that are crucial for developers to interact with the CORE8-16F framework effectively. Below are the key files and their purposes:

- **`core16F.h`**: This is the primary interface header for the core framework, containing:
  - Function prototypes for initializing and managing core functionalities.
  - Core definitions and macros that simplify development and ensure consistent use of common patterns.
  - Inline documentation for each function, explaining its purpose, parameters, and return values.
- **`core16F_const.h`**: This file defines all the constants used within the core system:
  - Constants such as clock frequency, system status codes, and other fixed values that maintain uniformity throughout the framework.
  - It ensures that all core modules use the same references, reducing discrepancies and simplifying debugging.
- **`core16F_init.c`**: Contains functions used for initializing the core framework:
  - **`void core_init(void)`**: This function sets up the initial configurations, including clock settings and system timers.
  - **Initialization Routines**: Initializes critical subsystems such as GPIO, timers, and interrupts. Each of these routines is modular, allowing developers to include or exclude components based on their project requirements.
- **`core_version.h`**: Used for tracking the version of the core:
  - Contains macros that specify the version number of the framework.
  - Useful for ensuring compatibility when integrating different versions of the core with other modules or projects.

Device Configuration (`device/`)

- **Device Configuration Files**: Contains configuration files specific to different PIC microcontrollers, such as `16F15313_configBits.h` and `16F1532x_configBits.h`, which provide settings for configuration bits like clock sources and watchdog timers.
- **Device-Specific Headers**: Files like `16F15313_core16F_config.h` and `16F1532x_core16F_config.h` contain setup information that is specific to each microcontroller

model.

Driver Directory (`drivers/`)

- **`ds18b20/`**: Contains `ds18b20.c` and `ds18b20.h` files for interfacing with the DS18B20 temperature sensor.
- **`lcd_i2c/`**: Provides the driver for I2C-connected LCD displays, including source (`lcd_i2c.c`) and header (`lcd_i2c.h`) files.
- **`led_rgb/`**: Handles RGB LED control with `led_rgb.c` and `led_rgb.h`, providing functions to adjust colors and brightness.

Hardware Abstraction Layer (`hal/`)

- **`gpio/`**: Manages GPIO pins with `gpio.c`, `gpio.h`, as well as analog-specific functionality (`gpio_analog.c`, `gpio_analog.h`). Provides functions for configuring and manipulating digital and analog pins.
- **`i2c1/`**: I2C protocol support (`i2c1.c`, `i2c1.h`), allowing communication with devices on the I2C bus.
- **`nco1/`**: Provides functionality to handle Numerically Controlled Oscillators (NCO) with `nco1.c` and `nco1.h`.
- **`one_wire/`**: Implements the One-Wire communication protocol with `one_wire.c` and `one_wire.h`, typically used for sensors like DS18B20.
- **`pps/`**: Manages the Peripheral Pin Select (PPS) feature (`pps.c`, `pps.h`) which allows flexible I/O mapping.
- **`pwm3/`**, **`pwm4/`**, **`pwm5/`**, **`pwm6/`**: PWM modules (`pwmX.c`, `pwmX.h`) providing functions to control Pulse Width Modulation outputs for different channels.
- **`serial1/`**: UART communication support (`serial1.c`, `serial1.h`, `serial1_isr.c`), handling serial communication including interrupt-based transmission and reception.
- **Timers (`tmr0/`, `tmr1/`, `tmr2/`)**: Provides timer support (`tmrX.c`, `tmrX.h`) for controlling time-based events and functions.

Interrupt Service Routines (`isr/`)

- **`isr_control.c`**, **`isr_control.h`**: Manages interrupt control, providing a central place for handling various hardware interrupts.
- **`main_isr.c`**: The main interrupt service routine, coordinating different interrupts such as timers, serial communication, and more.

Purpose and Use

The structure of the CORE8-16F framework is designed to provide modular and reusable components, allowing developers to easily add, modify, or remove features as needed. By understanding the purpose of each directory and its contents, developers can efficiently navigate the codebase and make effective contributions or adaptations to their specific needs.