Software Design

Serial Checker (SC0002)

“Details about the device’s embedded software architecture/design”

Rev A
12-JUL-2024

Purpose

The purpose of this document is to describe the software design of the Serial Checker device. This document should be able to provide insight to the overall architecture and design of the firmware.

Abbreviations

Abbreviation	Description

-	-

References

Name	Document Number
-	-

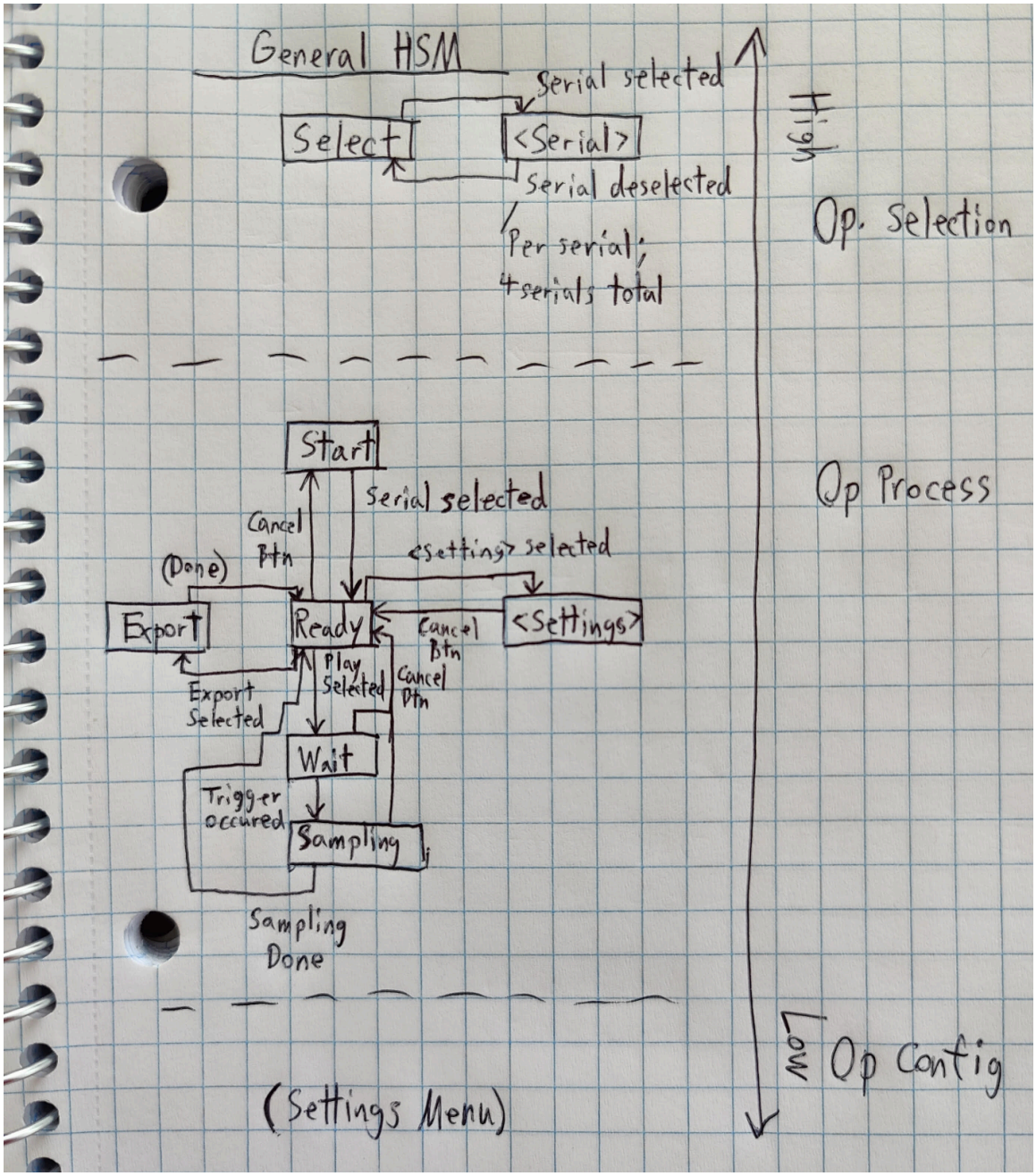
State Machine

This section describes how the firmware's state machine is organized and designed.

State Hierarchy

The firmware shall use a "hierarchical" state machine model. This model involves implementing the state machine in layers, where higher layered states determine larger portions of the program's behavior (and often determine if lower level state machines are even considered in decision making).

The firmware implements 3 layers: a serial operation layer, a serial process layer, and serial settings layer (ordered highest to lowest- see image below). The serial operation layer determines which serial type is being checked (and, by extension, which serial's data/info/settings is used). The serial process layer is used to track the core process of sampling and displaying data. The serial settings layer is used to track the user's movements through the various settings menus (note that these states do not affect the upper layers, but data selected within these states do).



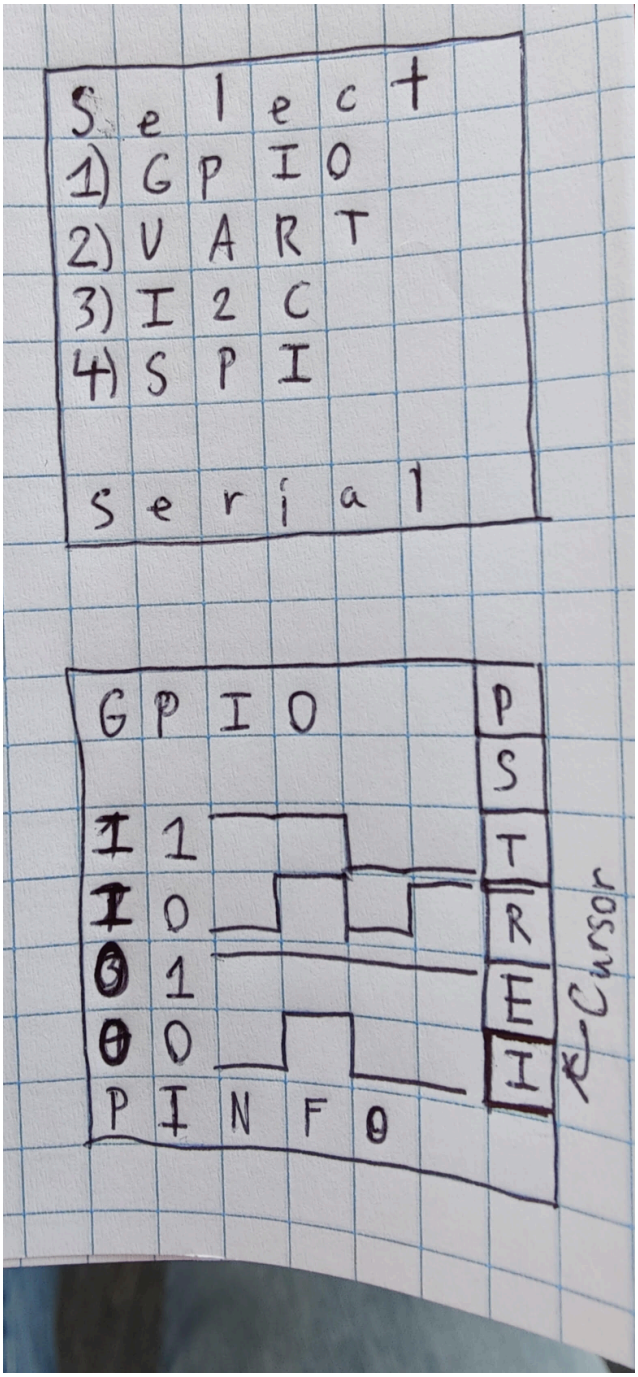
- Image 1: Hierarchical State Machine

LCD Screen

This section focuses on details related to the LCD screen's usage and programming.

Organization

The LCD screen provides 132x132 RGB pixels of resolution. For simplicity, the screen is divided into a 7 x 7 grid, where each grid piece can be given a custom border and central symbol (each grid piece being roughly 13 pixels square for the symbol, 17 pixels square with a border).



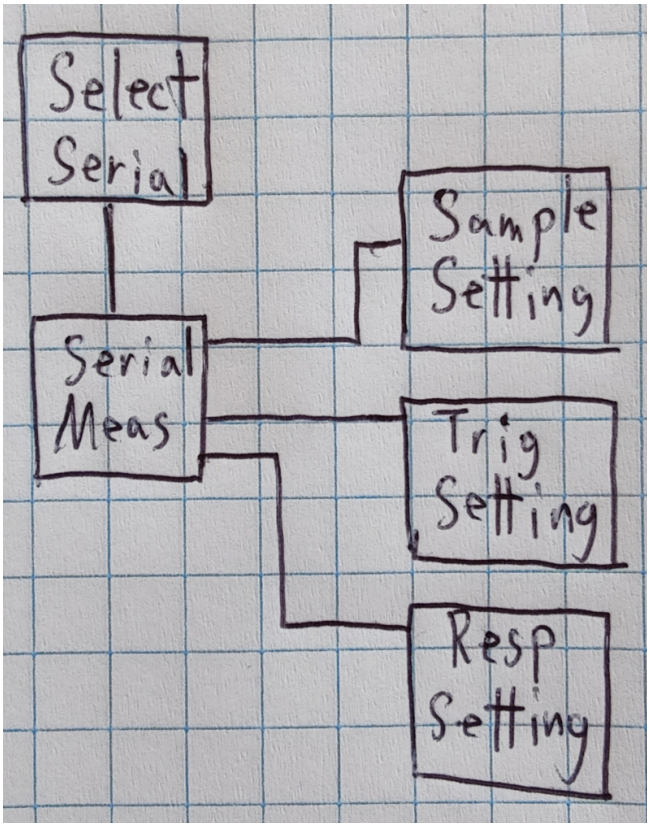
- Image 4: 7 x 7 Grid LCD Screen

User Interface Design

This section describes the user interface design. It primarily focuses on the displayed windows on the LCD screen.

Window Organization

The windows for the device are generally organized around the “Measurement” window. Once a serial is selected from the starting window, other windows (e.g. sampling/trigger/response settings) are accessed as options from the “Measurement” window (see image below).



- Image 3: Window Architecture

Settings Options

Most settings have preconfigured options that rely on the specific serial interface. Below is a summary of the options (per their serial interface).

Sample Settings

Sample settings are agnostic to the serial interface checked, but are still configured with predetermined options (and through the serial interface's measurement window, for consistency with the other settings).

Sample Rate (Hz) = {10, 50, 100, 500, 1000}
Sample Size (#) = {10, 50, 100, 500, 1000, 2000}

Trigger Settings

Trigger settings vary depending on the serial interface being checked. In general, triggers revolve around key input events triggering sampling/responses. User input can also act as a trigger.

Sample Trigger:

- GPIO = {"Confirm" Button Press, Pin 1 Rising, Pin 1 Falling, Pin 2 Rising, Pin 2 Falling}
- UART = {"Confirm" Button Press, TX Pin Rising, TX Pin Falling}
- I2C = {"Confirm" Button Press, SCL Pin Rising, SCL Pin Falling}
- SPI = {"Confirm" Button Press, CS Pin Rising, CS Pin Falling}

Response Trigger:

- GPIO = {"Confirm" Button Press, Pin 1 Rising, Pin 1 Falling, Pin 2 Rising, Pin 2 Falling, Off}
- UART = {"Confirm" Button Press, TX Pin Rising, TX Pin Falling, Receive Byte}
- I2C = {On, Off}
- SPI = {On, Off}

Note that for sampling, the first recorded sample is considered the sample just before the sample causing the trigger. This allows the trigger event to be observed in the sample (which is desirable from a data analysis perspective).

Response Settings

Serial interfaces often involve two way communication. For posterity, the serial checker should provide a way to simulate the serial on top of sampling it. Two response fields are present for each serial interface, though how they are used varies between the interfaces.

Response Field 1:

- GPIO = {Pin 3 Rises, Pin 3 Falls}
- UART = {Send Lower Byte, Send Both Bytes}
- I2C = <16-bit Address in Hex- set per hex digit in special screen>
- SPI = {Hold Low, Hold High, Output Data on Repeat}

Response Field 2:

- GPIO = {Pin 4 Rises, Pin 4 Falls}
- UART = <16-bit Value in Hex- set per hex digit in special screen>
- I2C = <16-bit Read Value in Hex- set per hex digit in special screen>
- SPI = <16-bit Out Value in Hex- set per hex digit in special screen>

Revision History

Rev	Date	Description of Changes	Initials
A	12-JUL-2024	Initial draft, mainly throwing ideas on a document	J.E.