HP-41C Dual Console Setup Guide

Quick Fixes Applied

1. Compilation Warning Fix

The unused `was_enabled` variable in `calculator.rs` has been removed:

```
rust

// OLD (had warning):
pub fn toggle_logging(&mut self) -> Option<String> {
    let was_enabled = self.logger.enabled;  // <-- unused variable
    let now_enabled = self.logger.toggle_enabled();
    Some(format!("Logging {}", if now_enabled { "ON" } else { "OFF" }))
}

// NEW (warning fixed):
pub fn toggle_logging(&mut self) -> Option<String> {
    let now_enabled = self.logger.toggle_enabled();
    Some(format!("Logging {}", if now_enabled { "ON" } else { "OFF" }))
}
```

2. File Logging Added

The logger now supports writing to both console AND file simultaneously:

- ✅ Console output (as before)
- ✅ File output (new feature)
- ✅ Automatic log file creation with headers
- ✅ Robust error handling for file operations

Dual Console Workflow

Terminal 1: Run the Calculator

```
bash

cd ~/hp41c
cargo run
```

Terminal 2: Tail the Log File

```bash
cd ~/hp41c
tail -f hp41c_debug.log
```

New Keyboard Controls

In the Calculator (Terminal 1):

Key Combo	Action
`Ctrl+F`	**Enable file logging** to `hp41c_debug.log`
`Ctrl+D`	**Disable file logging**
`Ctrl+L`	Toggle logging on/off
`Ctrl+A`	Enable ALL logging categories
`Ctrl+M`	Enable minimal logging (flags + stack)
`Ctrl+O`	Turn OFF all logging
`L`	Same as Ctrl+L (toggle logging)

Workflow Example:

1. **Start calculator:**

   ```bash
   cargo run
   ```

2. **In another terminal, prepare to tail:**

   ```bash
   tail -f hp41c_debug.log
   ```

   (This will wait for the file to be created)

3. **In calculator, press `Ctrl+F`**
   - Enables file logging
   - Creates `hp41c_debug.log`
   - Shows message: "File logging enabled: hp41c_debug.log"
   - Also shows: "You can now run: tail -f hp41c_debug.log"

4. **The tail window will now show real-time logs!**

5. **Press `Ctrl+A` to enable all logging categories**

6. **Start using the calculator** - every keystroke, stack operation, command, etc. will appear in the tail window

Log File Format

The log file includes:

- Session headers with timestamps
- Categorized log entries: `[INPUT]`, `[STACK]`, `[CMD]`, `[FLAG]`, `[PRGM]`, `[STORAGE]`
- Stack state before/after operations
- Command parsing details
- Flag changes
- Storage operations

Example log output:

```
=== HP-41C Calculator Log Session Started ===

[INPUT] Key: '5'
[INPUT] State: entering=true, eex=false, display='5_'
[STACK] digit_entry: T:    0.0000 Z:    0.0000 Y:    0.0000 X:    5.0000
[INPUT] Key: 'enter'
[STACK] ENTER operation
[STACK] Operation: enter
[STACK]   Before: T:    0.0000 Z:    0.0000 Y:    0.0000 X:    5.0000
[STACK]   After:  T:    0.0000 Z:    0.0000 Y:    5.0000 X:    5.0000
[INPUT] Key: '3'
[STACK] digit_entry: T:    0.0000 Z:    0.0000 Y:    5.0000 X:    3.0000
[INPUT] Key: '+'
[CMD] Execute: + -> completed
[STACK] Operation: + command
[STACK]   Before: T:    0.0000 Z:    0.0000 Y:    5.0000 X:    3.0000
[STACK]   After:  T:    0.0000 Z:    0.0000 Y:    0.0000 X:    8.0000
```

Code Changes Required

You'll need to apply the code changes from the artifacts above:

1. **Replace the `toggle_logging` method** in `calculator.rs`
2. **Replace the entire `logger.rs`** with the enhanced version
3. **Add the new methods** to `calculator.rs` for file logging support

4. **Update the logging method calls** that now need `&mut self`
5. **Replace** `main.rs` with the version that has file logging controls

Benefits

- 🎯 **Dual monitoring**: Calculator in one window, logs in another
- 🔍 **Real-time debugging**: See exactly what's happening as you type
- 📁 **Persistent logs**: Keep logs for later analysis
- ⚡ **Non-intrusive**: File logging doesn't slow down the calculator
- 🎛️ **Granular control**: Enable/disable specific log categories
- 🚀 **Easy setup**: Just `Ctrl+F` to start logging

This setup gives you the professional debugging experience you're looking for!