MC68705P3 Microcontroller Firmware Emulator

A precise C implementation that emulates the behavior of a Motorola MC68705P3 microcontroller firmware, specifically designed for the ND-120 minicomputer operator panel controller.

Overview

This emulator recreates the exact functionality observed in the disassembled Ghidra output from a 28-pin DIP MC68705P3 microcontroller. The firmware implements an intelligent peripheral controller that manages:

- Complete operator panel interface for ND-120 minicomputer system
- Real-time display management with LCD controller
- Button processing and debouncing
- Bidirectional CPU communication via CY7C401 FIFO
- 192-bit serial data packet reception and processing
- Complex command processing with dual lookup tables

Hardware Configuration

MC68705P3 Package: 28-pin DIP

- **VBB (Pin 1)**: -5V (Negative supply for EPROM programming)
- **VSS (Pin 14)**: 0V (Ground reference)
- **VDD (Pin 28)**: +5V (Main logic power supply)
- **XTAL (Pin 15)**: 2MHz Crystal oscillator

Port Configuration

- **PORTA (PA0-PA7, Pins 16-23)**: ALL INPUTS
  - Command reception from CPU
  - Button input monitoring
  - Serial data channels (PA0-PA2, active low)
  - Panel lock key detection (PA3)
- **PORTB (PB0-PB7, Pins 6-13)**: ALL OUTPUTS
  - Display data output
  - CPU response transmission
  - 7-bit data with status integration
- **PORTC (PC0-PC7, Pins 2-5, 24-27)**: ALL OUTPUTS

    - Control signals and timing
    - Serial sampling clock (PC1)
    - Display strobe signals (PC0)

System Integration

Communication Path

```
ND-120 CPU → PANC Register → CY7C401 FIFO → External Logic → 68705P3
```

Serial Protocol

- **Packet Size**: 192 bits (8 bytes × 3 channels × 8 bits)
- **Channels**:
    - Channel 1 (PA0): Time data
    - Channel 2 (PA1): Display data
    - Channel 3 (PA2): Status data
- **Clock**: Generated by PC1 toggle
- **Data Format**: Active-low inputs with MSB-first reception

Timing Coordination

- Synchronized with ND-120 20ms interrupt cycle
- CY7C401 FIFO (512×9-bit) buffers CPU commands
- Timer coordination maintains protocol timing

Build Instructions

Prerequisites

- GCC compiler (C99 or newer)
- Make utility
- Optional: Valgrind, cppcheck, doxygen

Basic Build

```bash
make all          # Build emulator and test suite
make run          # Build and run emulator
make test         # Build and run test suite
```

Development Build

```bash
make debug          # Debug build with AddressSanitizer
make analyze        # Static analysis with cppcheck
make valgrind       # Memory checking with Valgrind
```

Release Build

```bash
make release        # Optimized release build
make install        # Install to system directory
```

Usage

Running the Emulator

```bash
./mc68705_emulator
```

Running Tests

```bash
./mc68705_test
```

Example Output

```
MC68705P3 ND-120 Operator Panel Controller Emulator
Hardware: 28-pin DIP, 2MHz Crystal, CY7C401 FIFO Interface
Initializing hardware...
Display Command: 0x38
Display Command: 0x0C
Display Command: 0x06
Display Clear Pulse
...
```

API Reference

Core Functions

`void RESET(void)`

Address: 0x00D6
Hardware initialization and system startup. Configures all ports, initializes timer, sets up display system, and begins main processing loop.

`void WaitForData(void)`

Address: 0x025E
CPU command reception via hardware interface. Handles 192-bit serial data reception from ND-120 CPU with proper timing coordination.

`uint8_t ProcessData(void)`

Address: 0x0142
Main command processing state machine. Processes 6-bit commands from PORTA using dual lookup table system with 128+ dispatch codes.

Command Categories

Command Range	Function	Description
0x00-0x02	Display Updates	Complex multi-stage display operations
0x08-0x0C	Button Polling	Button input with debouncing
0x48-0x4A	Direct Output	Raw data transmission
0x77-0x7F	Serial Reception	192-bit packet processing

Display Functions

`void DisplayTimeData(uint8_t buffer_base)`

Displays time information from specified buffer with automatic character lookup and binary digit handling.

`void SendDisplayCommand(uint8_t cmd)`

Sends control commands to LCD display controller.

`void OutputCharacterToDisplay(uint8_t character)`

Outputs single character to display with position management.

Memory Layout

RAM Structure (0x0010-0x004C)

```c
typedef struct {
    uint8_t control_flags[0x15];       // 0x0010-0x0024: Control/status
    uint8_t message_buffers[8];        // 0x0025-0x002C: Message storage
    uint8_t time_data_buffer[8];       // 0x002D-0x0034: Raw time data
    uint8_t time_display_buffer[8];    // 0x0035-0x003C: Display time
    uint8_t status_data_buffer[8];     // 0x003D-0x0044: System status
    uint8_t additional_vars[8];        // 0x0045-0x004C: Additional state
} System_RAM;
```

ROM Constants (0x0080-0x077F)

- **0x0080-0x008D**: Command lookup tables (primary/secondary)
- **0x0098-0x00D5**: Display strings ("ON", "OFF", "TIME:", etc.)
- **0x06BC-0x077F**: Character decode and lookup tables

Architecture Details

Command Processing Flow

1. **Input Sampling**: PORTA monitored for 6-bit commands
2. **Debouncing**: 5-cycle debounce counter prevents false triggers
3. **Table Lookup**: Dual lookup system based on display flags
4. **Dispatch**: 128+ case switch statement for command execution
5. **Response**: 7-bit output via PORTB with strobe signaling

Serial Data Reception

1. **IRQ Detection**: Hardware signal indicates data availability
2. **Clock Generation**: PC1 toggle creates sampling clock
3. **Bit Shifting**: Active-low inputs shifted into 3 parallel registers
4. **Packet Assembly**: 8 bytes per channel stored in dedicated buffers
5. **Processing**: Character decoding and display formatting

Timer System

- **DAT_0010**: Primary countdown timer
- **DAT_0011**: Secondary countdown timer
- **DAT_0013**: Debounce counter (5 cycles)
- **Special Values**: 0x50 (timer coordination), 0x06 (display refresh)

Testing

The emulator includes a comprehensive test suite covering:

- Hardware initialization verification
- Port configuration testing
- Command processing validation
- Serial data reception testing
- Display function verification
- Button handling and debouncing
- Timer operations and coordination
- Character decoding accuracy
- Lookup table functionality
- Error condition handling
- Integration scenarios

Test Results Example

```
MC68705P3 Microcontroller Emulator Test Suite
==============================================

--- Running Test: Hardware Initialization ---
PASS: Port B configured as output
PASS: Port C configured as output
PASS: Port A configured as input
PASS: Timer control register initialized
--- Test Completed Successfully ---

Test Results Summary:
Tests Run: 11
Tests Passed: 11
Tests Failed: 0
OVERALL RESULT: PASSED
```

Simulation Features

Hardware Interface Simulation

```c
// Simulate button press
simulate_button_press(0x21);

// Simulate CPU command
simulate_cpu_command(0x10);

// Simulate serial data reception
uint8_t time_data[8] = {0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC, 0xDE, 0xF0};
uint8_t disp_data[8] = {0x20, 0x31, 0x32, 0x3A, 0x33, 0x34, 0x20, 0x20};
uint8_t stat_data[8] = {0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
simulate_serial_data(time_data, disp_data, stat_data);
```

Debug Output

The emulator provides detailed debug output showing:

- Port state changes
- Command processing steps
- Display operations
- Timer updates
- Buffer contents
- System statistics

Technical Specifications

Performance Characteristics

- **Clock Speed**: 2MHz crystal oscillator
- **Instruction Cycle**: ~500ns typical
- **Response Time**: <50μs for command processing
- **Serial Data Rate**: Synchronized with 20ms CPU interrupt
- **Display Update Rate**: Variable based on content changes

Memory Requirements

- **ROM**: 2KB (0x0000-0x07FF)
- **RAM**: 64 bytes (0x0010-0x004F)
- **Registers**: 12 bytes (0x0000-0x000B)
- **Total**: ~2.1KB firmware image

Electrical Specifications

- **Supply Voltage**: +5V ±5%
- **Logic Levels**: TTL compatible
- **Input Current**: <1mA per pin
- **Output Current**: 20mA source/sink capability
- **Power Consumption**: ~50mW typical

Protocol Documentation

Command Structure

```
PORTA Bit Assignment:
 7   6   5   4   3   2   1   0
[S] [B] [---- Command ----]

S = Status/control bit
B = Button change flag
Command = 6-bit command code (0x00-0x3F)
```

Response Format

```
PORTB Bit Assignment:
 7   6   5   4   3   2   1   0
[S] [------ Data ------]

S = Status integration bit
Data = 7-bit response data
```

Display Commands

Command	Function	Description
0x38	Function Set	8-bit interface, 2-line, 5×7 font
0x0C	Display On	Display on, cursor off, blink off
0x06	Entry Mode	Increment cursor, no display shift
0x01	Clear Display	Clear entire display
0x02	Return Home	Return cursor to home position
0x18	Shift Left	Shift display content left

Troubleshooting

Common Issues

Emulator Won't Start

- Check compiler version (requires C99 or newer)
- Verify all source files are present
- Run `make clean && make all`

Test Failures

- Ensure clean build environment
- Check for memory leaks with `make valgrind`
- Run static analysis with `make analyze`

Display Output Issues

- Verify character lookup tables are correct
- Check display command sequence
- Ensure proper timing coordination

Serial Reception Problems

- Confirm IRQ status simulation
- Verify bit shifting logic
- Check buffer boundary conditions

Debug Mode

```bash
make debug
./mc68705_emulator_debug
```

Enables:

- AddressSanitizer for memory error detection
- Enhanced debug output
- Assertion checking
- Stack trace on errors

Contributing

Development Setup

1. Fork the repository
2. Create feature branch
3. Make changes with tests

4. Run full test suite
5. Submit pull request

Code Style

- Follow C99 standard
- Use descriptive variable names
- Comment complex logic sections
- Maintain consistent indentation
- Add tests for new features

Testing Requirements

- All new features must include tests
- Maintain 100% test coverage
- Performance tests for timing-critical code
- Integration tests for hardware interfaces

License

This emulator is provided for educational and research purposes. The implementation is based on reverse engineering of the original MC68705P3 firmware through disassembly analysis.

Acknowledgments

- Based on Ghidra disassembly analysis
- MC68705P3 datasheet reference
- ND-120 system documentation
- CY7C401 FIFO interface specifications

References

Technical Documentation

- Motorola MC68705P3 Microcontroller Data Sheet
- ND-120 Computer System Technical Manual
- CY7C401 FIFO Memory Data Sheet
- HD44780 LCD Controller Command Reference

Development Tools

- Ghidra - NSA's reverse engineering framework

- GCC - GNU Compiler Collection
- Valgrind - Memory debugging tool
- Cppcheck - Static analysis tool

Changelog

Version 1.0.0

- Initial implementation based on Ghidra disassembly
- Complete hardware register emulation
- Full command processing state machine
- Serial data reception protocol
- Display management system
- Comprehensive test suite
- Documentation and build system

Future Enhancements

- Real-time visualization of display output
- Logic analyzer integration for timing analysis
- Hardware-in-the-loop testing capability
- Performance profiling and optimization
- Extended command set simulation
- Network interface for remote operation

Contact

For questions, bug reports, or contributions, please refer to the project repository or contact the development team.

This emulator provides a faithful reproduction of the MC68705P3 firmware behavior as observed through disassembly analysis. All functionality is implemented based strictly on the observable code patterns without speculation or invention of features not present in the original firmware.