# Hardware Documentation

## Table of Contents

# Introduction

Quadium 4004 Workbench simulates a complete Intel MCS-4 microprocessor system, including the 4004 CPU, 4001 ROM chips, 4002 RAM chips, and 4003 shift register chips. The system provides full hardware emulation with support for I/O ports, peripheral connections, and real-time execution.

**Main components:**

- **CPU 4004**: 4-bit microprocessor with 12-bit program counter
- **ROM 4001**: 256-byte read-only memory with 4-bit I/O port
- **RAM 4002**: 64-nibble data memory + 4-nibble status memory with 4-bit I/O port
- **Shift Register 4003**: 10-bit serial shift register
- **MCS-4 Bus**: Interconnect bus managing all chip communications

**Default topology:**

- 16 × 4001 ROM chips (4096 bytes total)
- 16 × 4002 RAM chips (1024 nibbles data + 64 nibbles status total)
- 3 × 4003 Shift Register chips (30 bits total)

**Interconnect diagram:** Help → **MCS-4 interconnect diagram (Workbench)** opens a single window with two views of the same emulation model, toggled with **Modern** / **Vintage**.
*Modern* shows the high-level block diagram of how this emulator ties the CPU, ROM/RAM, fixed 4003 chains, and user peripherals together.
*Vintage* shows a “electronics sheet” style drawing.
Neither view is Intel reference artwork—both are original Quadium documentation art for this product.

Vintage-style system sheet (Intel-manual inspired layout, **original Quadium artwork**): `assets/mcs4-quadium-vintage-interconnect.svg` (Obsidian: `Manuales/assets/`). Detailed code topology: `assets/mcs4-quadium-interconnect-schematic.svg`.

# CPU 4004

This section describes the emulated 4004 microprocessor architecture. The 4004 is a 4-bit CPU with the following architectural features:

## Registers and State

| Component | Size | Range | Description |
|---|---|---|---|
| PC | 12 bits | 0..4095 | Program Counter |
| ACC | 4 bits | 0..15 | Accumulator |
| CY | 1 bit | 0/1 | Carry/Link flag |
| R0..R15 | 16 × 4 bits | 0..15 each | General-purpose registers |
| L1..L3 | 3 × 12 bits | 0..4095 each | 3-level subroutine stack |
| DCL Bank | 4 bits | 0..15 | RAM bank selector (`DCL` / opcode FD; low nibble of ACC in this emulator) |
| ROM I/O Chip | 4 bits | 0..15 | Selected ROM chip for I/O (SRC) |

## Instructions

The emulator implements the complete 4004 instruction set, including:

- **Arithmetic/Logic**: ADD, SUB, ADD with Carry, AND, OR, XOR, etc.
- **Register Operations**: LD, XCH, INC, DEC
- **Memory Access**: WRM, RDM, WRn, RDn, WMP, WRR, RDR, etc. (`RMP` is not executed by the emulated CPU)
- **Status**: WR0..WR3, RD0..RD3 (4002 status registers)
- **Control Flow**: JCN, JUN, JMS, BBL, FIM, FIN, SRC, JIN
- **Special**: DCL (set RAM bank), KBP, DAA, CLC, STC, CMC

## Stack

The 4004 has a 3-level hardware stack for subroutine calls:

- **L1**: Top of stack (most recent return address)
- **L2**: Second level (if depth ≥ 2)
- **L3**: Bottom level (if depth = 3)

**Stack depth**: 0..3 (0 = empty, 3 = full)

# ROM 4001

This section describes the emulated 4001 ROM chip with integrated 4-bit I/O port.

## Memory Organization

| Component | Size | Range | Description |
|---|---|---|---|
| ROM Page | 256 bytes | 0..255 | Read-only program/data memory |
| I/O Port | 4 bits | 0..15 | Bidirectional I/O port |

## Addressing

ROM chips are addressed by a 12-bit address:

- **Upper 4 bits (12:9)**: Select ROM chip (0..15)
- **Lower 8 bits (8:0)**: Select byte within chip (0..255)

**Address calculation:**

```
chipAbs = (address >> 8) & 0x0F  // 0..15
offset  = address & 0xFF         // 0..255
```

## I/O Port

The 4001 I/O port is a 4-bit bidirectional port:

- **WRR**: Write accumulator to ROM port
- **RDR**: Read ROM port to accumulator

**Port access:**

- Chip selection via **SRC** instruction (nibble in register pair)
- Write: `WRR` writes ACC to selected ROM chip's port
- Read: `RDR` reads selected ROM chip's port to ACC

# RAM 4002

This section describes the emulated 4002 RAM chip with data memory, status memory, and I/O port.

## Memory Organization

| Component | Size | Range | Description |
|---|---|---|---|
| Data RAM | 4 × 16 nibbles | Reg 0..3, Digit 0..15 | Main data storage |
| Status RAM | 4 nibbles | Status 0..3 | Special-purpose status registers |
| I/O Port | 4 bits | 0..15 | Write-only I/O port |

## Addressing

RAM chips are organized in banks (DCL) with relative chip selection:

### Bank selection (DCL):

- Set via DCL instruction (opcode FD): stores the low nibble of ACC as the current RAM bank (0..15 in this emulator).
- **Standard MCS-4 with 16 × 4002:** four chips per bank → banks **0..3** cover all installed RAM (`chipAbs = bank × 4 + chipRel`, chips 0..15). That is the normal full configuration, not a reduced mode.
- Bank numbers **4..15** do not address another chip in a 16-chip system. If software selects them anyway, this emulator behaves like open bus on an unpopulated decode: reads return **0**, writes are ignored.

### Chip selection (SRC):

- **SRC r**: Sets chip (0..3), register (0..3), and digit (0..15) within selected bank
- Register pair encoding:
  - `chip = (pair8 >> 6) & 0x03` // 0..3
  - `reg = (pair8 >> 4) & 0x03` // 0..3
  - `digit = pair8 & 0x0F` // 0..15

### Absolute chip calculation:

```
chipAbs = (DCL_bank * 4) + chipRel  // 0..15
```

## Data RAM

- **WRM**: Write ACC to selected data RAM nibble
- **RDM**: Read selected data RAM nibble to ACC
- **WRn**: Write ACC to status register n (0..3)
- **RDn**: Read status register n (0..3) to ACC

## Status RAM

Four special-purpose 4-bit status registers per chip:

- **WR0/RD0**: Status register 0
- **WR1/RD1**: Status register 1
- **WR2/RD2**: Status register 2
- **WR3/RD3**: Status register 3

## I/O Port

The 4002 I/O port is **write-only** from the CPU:

- **WMP**: Write accumulator to RAM port
- Port bits: `DATA = bit0`, `CLK = bit1`

**Note**: Reading the RAM port (`RMP`) is not directly supported by the 4004 CPU instruction set, but the emulator allows reading for debugging purposes.

# Shift Register 4003

This section describes the emulated 4003 shift register, which is a 10-bit serial shift register.

## Organization

| Component | Size | Range | Description |
|---|---|---|---|
| Register | 10 bits | 0..1023 | Internal shift register |
| IN Bit | 1 bit | 0/1 | Last bit shifted in |
| OUT Bit | 1 bit | 0/1 | Last bit shifted out |

## Shift Chain

The system supports multiple 4003 chips in a daisy chain:

Chain A (ROM chip 15):

- 4003 #0 is clocked from ROM chip 15 port (bit1 = CLK, bit0 = DATA)
- Rising edge on CLK: shifts DATA bit into chain A

Chain B (RAM bank 3, chip 1 = chip absolute 13):

- 4003 #1 and #2 are clocked from RAM chip 13 port (bit1 = CLK, bit0 = DATA)
- Rising edge on CLK: shifts DATA bit through chain B (#1 → #2)

## Operation

### Clock operation:

- On rising edge of CLK, the register shifts left
- LSB receives the DATA input bit
- MSB is shifted out

### Shift function:

```
outBit = (register >> 9) & 1
register = ((register << 1) | inBit) & 0x3FF
```

# MCS-4 Bus and Chip Wiring

The MCS-4 bus is the physical interconnect that connects all chips in the system. Understanding the wiring is essential for understanding how the system works.

## Bus Architecture

The MCS-4 uses a **multiplexed bus** that carries:

- **Address/data**: 4-bit data bus, time-multiplexed
- **Control signals**: SYNC, CM (Command), CM-RAM (RAM Command), etc.
- **Chip selection**: Decoded from address/command lines

## CPU to Bus Connection

The 4004 CPU connects to the bus via:

- **4-bit data bus**: Bidirectional, carries addresses and data
- **SYNC signal**: Synchronization clock
- **CM lines**: Command lines (CM0, CM1) indicate instruction phase
- **CM-RAM lines**: RAM command lines for 4002 chip control

## ROM Chip Wiring

**Physical connections:**

- **16 ROM chips** (4001) connect in parallel to the bus
- Each chip decodes its own **chip select** from the upper address bits (A8-A11)
- **ROM chips 0-15**: Selected when upper address nibble matches chip number

**I/O Port wiring:**

- Each 4001 has a **4-bit I/O port** that can be read/written independently
- Port values are latched internally in each chip
- **Port selection**: Via `SRC` instruction, which sets which ROM chip's port is active

**Special ROM chip connections:**

- **ROM chip 15**: Port bit 0 (DATA) and bit 1 (CLK) connect to **4003 shift register #0**
- **ROM chips 0, 1, 2**: Used for byte output protocol (see I/O Ports section)

## RAM Chip Wiring

**Physical connections:**

- **16 RAM chips** (4002) organized in **4 banks** of 4 chips each
- **DCL bank selection**: Set via `DCL` instruction (lines DCL0-DCL1)
- **Chip selection within bank**: Via `SRC` instruction (upper 2 bits of register pair)
- Each bank contains 4 chips (0-3 within that bank)

**Bank organization:**

- **Bank 0**: RAM chips 0, 1, 2, 3
- **Bank 1**: RAM chips 4, 5, 6, 7
- **Bank 2**: RAM chips 8, 9, 10, 11
- **Bank 3**: RAM chips 12, 13, 14, 15

**QuadBasic system RAM:** When you use the QuadBasic transpiler, the **last RAM chip** (bank 3, chip 3 — absolute chip 15) reserves **registers 2 and 3** (32 four-bit data cells) for the compiler and runtime. Registers 0 and 1 on that chip may still hold user variables in very large programs. This is unrelated to the shift-register port on bank 3, chip 1 (absolute chip 13). R0..R9 hold user `NIBBLE` variables; R10..R15 and cells `RAM[3, 3, 3, 0..15]` are system use.

**Absolute chip calculation:**

```
chipAbs = (DCL_bank × 4) + chipRel
```

**I/O Port wiring:**

- Each 4002 has a **4-bit I/O port** (write-only from CPU)
- Port bits: **bit 0 = DATA**, **bit 1 = CLK** for shift register chains
- **RAM chip 13** (Bank 3, chip 1): Port connects to **4003 shift registers #1 and #2**

## Shift Register Chain Wiring

The system supports two independent shift register chains:

**Chain A (ROM-based):**

- **Source**: ROM chip 15 port
- **Connection**: Port bit 0 (DATA) → 4003 #0 input
- **Clock**: Port bit 1 (CLK) → 4003 #0 clock (rising edge)
- **Chain**: 4003 #0 only (or can extend to more chips)

**Chain B (RAM-based):**

- **Source**: RAM chip 13 port (Bank 3, chip 1, absolute chip 13)
- **Connection**: Port bit 0 (DATA) → 4003 #1 input
- **Clock**: Port bit 1 (CLK) → 4003 #1 and #2 clock (rising edge)
- **Chain**: 4003 #1 → 4003 #2 (serial daisy chain)

**Shift register operation:**

- On **rising edge of CLK**, data shifts left
- **LSB** receives the DATA input bit
- **MSB** is shifted out to the next chip (if chained)

## Address Decoding

**ROM addressing:**

- **12-bit address** (A0-A11)
- **A8-A11** (upper 4 bits): Select ROM chip (0-15)
- **A0-A7** (lower 8 bits): Select byte within chip (0-255)

**RAM addressing:**

- **Multi-level selection**:
  1. **DCL** (A0-A1): Selects bank (0-3)
  2. **SRC** upper nibble (bits 6-7): Selects chip within bank (0-3)
  3. **SRC** middle nibble (bits 4-5): Selects register (0-3)
  4. **SRC** lower nibble (bits 0-3): Selects digit/nibble (0-15)

## Port Latches

Each chip maintains its own **port latch**:

- **ROM ports**: One latch per ROM chip (16 latches total)
- **RAM ports**: One latch per RAM chip (16 latches total)

**Purpose of latches:**

- **Edge detection**: Detect rising edges for shift register clocks
- **Port state retention**: Maintain port values between operations
- **Snapshot support**: Allow saving/restoring complete system state

# I/O Ports

## ROM Ports

### Architecture:

- 16 ROM chips, each with a 4-bit I/O port
- Bidirectional: CPU can read and write
- External peripherals can inject values (OR-wired)

### CPU Access:

- **Write**: `WRR` instruction writes ACC to selected ROM chip port
- **Read**: `RDR` instruction reads selected ROM chip port to ACC
- **Selection**: ROM chip selected via `SRC r` (high nibble of register pair)

### Special ROM Ports:

| Chip | Usage | Description |
|---|---|---|
| **ROM 0** | STROBE | Bit 0 = strobe signal (edge-triggered byte output) |
| **ROM 1** | HI nibble | High nibble of byte being output |
| **ROM 2** | LO nibble | Low nibble of byte being output |
| **ROM 15** | Shift Chain A | Bit 0 = DATA, Bit 1 = CLK for 4003 #0 |

### Byte Output Protocol:

1. CPU writes HI nibble to ROM 1 port
2. CPU writes LO nibble to ROM 2 port
3. CPU writes strobe bit (0→1) to ROM 0 port
4. System latches byte and routes to selected device

## RAM Ports

### Architecture:

- 16 RAM chips, each with a 4-bit write-only I/O port
- External peripherals can inject values (OR-wired)
- Used primarily for shift register chains

### CPU Access:

- **Write**: `WMP` instruction writes ACC to selected RAM chip port
- **Selection**: RAM chip selected via current DCL bank + `SRC` chip

### Port Bits:

- **Bit 0**: DATA input for shift register
- **Bit 1**: CLK input for shift register (rising edge triggers shift)

### Special RAM Port:

| Bank | Chip Rel | Chip Abs | Usage | Description |
|---|---|---|---|---|
| 3 | 1 | 13 | Shift Chain B | Bit 0 = DATA, Bit 1 = CLK for 4003 #1 and #2 |

# Peripherals

The system supports various user-configurable peripherals that can connect to I/O ports.

## Available Peripherals

| Peripheral ID | Name | Description | Size |
|---|---|---|---|
| ledbar | LED Bar | N LEDs; single bitmap channel | 1..64 elements |
| sevensegbcd | 7-Segment BCD | N 7-segment digits; 4-bit BCD per digit | 1..16 elements |
| sevensegseg | 7-Segment SEG | N 7-segment digits; per-digit segment mask | 1..16 elements |
| printer | Printer | Printer (32×8). Centronics-like protocol | Fixed (1) |
| lcd16x2 | LCD | LCD (16×2). DATA(hi,lo)+STROBE protocol | Fixed (1) |
| asciikeyboard | ASCII Keyboard | Full keyboard; writes ASCII to ROM ports | Fixed (1) |
| dip4 | 4-DIP Switch | 4-position DIP switch | Fixed (1) |
| pushbuttons | Push Button Strip | N push buttons | Variable |
| pulsegen | Pulse Generator | Programmable pulse generator | Fixed (1) |
| vdp64x48 | VDP 64×48 (16 colors) | TMS9918-style video (ROM DATA/CTRL write, STATUS read) | Fixed (1) |

## Peripheral Binding

Each peripheral has **binding templates** that define how it connects to the system:

**Binding Configuration:**

- **SourceKind**: Where to read/write (`CpuCore`, `Rom4001Port`, `Ram4002Port`, `Shift4003`)
- **PathKind**: What to read/write (`PC`, `ACC`, `SR`, `Nibble`, `Port`)
- **Chip**: Chip number (0..15 for ROM, bank+chip for RAM)
- **Reg**: Register/index (0..3 for RAM, 0..15 for CPU registers)
- **CharIndex**: Character/digit index (0..15 for RAM)
- **BitOrder**: Bit ordering (`LSB_TO_MSB`, `MSB_TO_LSB`)
- **Invert**: Invert logic levels

## Peripheral Directions

In the **User Peripherals** editor, **Input** vs **Output** refers to how the module interacts with its *binding source* (not "current into the chip" in an analog sense):

| Direction | Role in this workbench | Example |
|---|---|---|
| **Input** | The peripheral **reads** a readable source each snapshot (CPU core, ROM port latch, RAM port latch, shift register, …) and reflects it in the UI. | LED bar / seven-segment displays bound to `Ram4002Port` or `CpuCore`. |
| **Output** | The peripheral **drives** extra bits that are **OR-combined** with the selected **ROM 4001** port nibble before `RDR` (and cleared when idle). The binding UI only offers **Rom4001Port** for this direction today. | DIP switch, push buttons, ASCII keyboard, pulse generator → `Rom4001Port`. |
| **Bidirectional** | Reserved / same provider list as "either" in the UI filter. | Not used by stock catalog modules. |

The **board** also exposes `SetRamPortExternalInput` for future OR-injection on RAM port nibbles; the catalog's Output filter is still ROM-only in the editor.

# Memory Addressing

## ROM Addressing

**12-bit address space:**

- **Address range**: 0..4095 (0x000..0xFFF)
- **Chip selection**: `(address >> 8) & 0x0F` → ROM chip 0..15
- **Byte offset**: `address & 0xFF` → byte 0..255 within chip

**Example:**

```
Address 0x123:
  chipAbs = 0x1 = ROM chip 1
  offset  = 0x23 = byte 35 within chip
```

## RAM Addressing

**Multi-level addressing:**

1. **Bank**: Set via `DCL` (low nibble of ACC). With the **standard 16-chip RAM array**, banks **0..3** are the only bank values that select a physical chip; bank × 4 + chip (0..3) yields `chipAbs` 0..15.
2. **Chip**: Relative within bank (0..3), set via `SRC` high nibble
3. **Register**: Within chip (0..3), set via `SRC` middle nibble
4. **Digit**: Within register (0..15), set via `SRC` low nibble

**Absolute chip calculation:**

```
chipAbs = (DCL_bank * 4) + chipRel
```

**Example:**

```
DCL bank = 2
SRC sets: chip=1, reg=2, digit=5

chipAbs = 2*4 + 1 = 9  (RAM chip 9)
Access: RAM chip 9, register 2, digit 5
```

## Status RAM Addressing

### Per-chip status registers:

- Selected via current RAM bank + chip
- Status index: 0..3 (direct from instruction `WR0..WR3`, `RD0..RD3`)

### Example:

```
DCL bank = 1
SRC chip = 2
WR1 instruction:

chipAbs = 1*4 + 2 = 6  (RAM chip 6)
Write ACC to status register 1 of chip 6
```

# Connection Model

## OR-Wired Accumulative Model

When multiple peripherals connect to the same input port, their values are combined using **OR logic**, simulating a real **wired-OR** electrical bus.

**Electrical behavior:**

- In a real circuit, multiple devices can drive the same bus line
- If ANY device drives a line **HIGH**, the line reads as HIGH
- The line is **LOW** only if ALL devices drive it LOW (or are in high-impedance state)

**Simulated behavior:**

- **Effective value** = Internal chip value **OR** (Peripheral1 **OR** Peripheral2 **OR** ... **OR** PeripheralN)
- A bit is **high** if ANY source (internal chip or any peripheral) sets it high
- A bit is **low** only if ALL sources set it low

**Example:**

```
ROM chip 0 port connections:
  Internal chip value (from CPU write):    0b0011 (3)
  Peripheral A (4-DIP switch):             0b0101 (5)
  Peripheral B (Push button):              0b1010 (10)

When CPU reads the port:
  Effective read value = 0b1111 (15)
  = 0b0011 OR 0b0101 OR 0b1010

Explanation:
  Bit 0: 1 OR 1 OR 0 = 1
  Bit 1: 1 OR 0 OR 1 = 1
  Bit 2: 0 OR 1 OR 0 = 1
  Bit 3: 0 OR 0 OR 1 = 1
```

**Multiple peripherals:**

- Multiple peripherals can connect to the same port simultaneously
- Values are automatically OR-combined when the CPU reads the port
- This allows complex input configurations (e.g., multiple switches or buttons on one port)

# Emulator Features

The Quadium 4004 Workbench provides several features to help you work with the emulated MCS-4 system.

## Snapshots

### What are snapshots?

- Snapshots capture the **complete state** of the emulated system at a specific moment
- Includes: CPU registers, memory (ROM/RAM), ports, shift registers, status registers
- Snapshots are **immutable** and **atomic** (all or nothing)

### Snapshot uses:

- **Save state**: Save your program's current state to reload later
- **Debugging**: Capture state at breakpoints or errors
- **Testing**: Compare states before/after operations
- **Restore**: Restore the system to a previously saved state

### Snapshot contents:

- **CPU state**: PC, ACC, CY, all 16 registers (R0-R15), stack (L1-L3)
- **Memory**: All ROM bytes, all RAM data nibbles, all RAM status nibbles
- **Ports**: All ROM port values, all RAM port values
- **Shift registers**: All 4003 chip states (register values, last in/out bits)
- **Bus state**: Current DCL bank, selected chip/reg/digit
- **Metadata**: Sequence number, timestamp

### Snapshot operations:

- **Save**: Export current state to a file
- **Load**: Restore state from a file
- **Auto-save**: System automatically saves snapshots periodically (when running)

## Execution Speed (IPS)

**IPS (Instructions Per Second):**

- Controls how fast the CPU executes instructions
- Measured in **instructions per second**
- Can be set from very slow (for debugging) to very fast (for testing)

**Speed modes:**

- **Very slow**: 0.1 IPS (one instruction every 10 seconds)
- **Fixed IPS**: Execute at a constant rate (factory default **200 IPS** until user settings load)
- **Real MCS-4 speed**: Emulate actual 4004 CPU speed (~46–93 kIPS)
- **Turbo**: Execute as fast as possible (no IPS cap)

**Speed presets** (editable names; IPS values are defaults when no user config file exists):

- **Ultra slow**: 0.25 IPS
- **Very slow**: 0.5 IPS
- **Slow**: 1 IPS
- **Trace**: 2 IPS
- **Inspect**: 5 IPS
- **Low**: 10 IPS
- **Medium**: **25 IPS** (default preset name)
- **High**: 500 IPS
- **Very high**: 1000 IPS
- **Fast**: 2500 IPS (editable up to 5,000,000 IPS)

**Real MCS-4 speeds:**

- **4004 (2-cycle avg)**: ~46,300 IPS (average instruction timing)
- **4004 (1-cycle max)**: ~92,600 IPS (fastest instruction timing)

**IPS display:**

- Current IPS is displayed in real-time on the main window
- Updated every 500ms
- Shows actual execution rate (may vary due to system load)

## Snapshot Frequency

### What is snapshot frequency?

- Controls how often the UI updates with new snapshots
- Measured in **frames per second (FPS)**
- Higher frequency = smoother UI updates, but more CPU usage

### Default settings:

- **Default FPS**: 60 FPS (60 snapshots per second)

### Snapshot timing:

- Snapshots are captured **asynchronously** from execution
- Execution continues at full speed even if UI updates are slower
- UI always shows the **latest available snapshot**

## Breakpoints

### What are breakpoints?

- Pause execution when the program counter reaches a specific address
- Useful for debugging and inspection
- Multiple breakpoints can be set simultaneously

### Breakpoint behavior:

- When hit, execution **pauses** immediately
- System state is captured in a snapshot
- You can inspect registers, memory, ports, etc.
- Execution can be resumed (step-by-step or continuous)

## Examples

### Example 1: Reading ROM Port

```
; Select ROM chip 5 for I/O
FIM 0P, 0x50   ; R0=5, R1=0
SRC 0P         ; Select ROM chip 5

; Write to port
LDM 12
WRR            ; Write 12 (0xC) to ROM chip 5 port

; Read from port
RDR            ; Read ROM chip 5 port to ACC
```

### Example 2: RAM Access

```
; Set RAM bank 2
LDM 2
DCL            ; Set DCL bank = 2

; Select chip 1, register 3, digit 10
FIM 2P, 0x3A   ; R4=3, R5=10
FIM 1P, 0x13   ; R2=1, R3=3
SRC 2P         ; chip=1 (from R2), reg=3 (from R2), digit=10 (from R5)

; Write to data RAM
LDM 5
WRM            ; Write 5 to RAM bank 2, chip 1, reg 3, digit 10

; Read from data RAM
RDM            ; Read nibble to ACC
```

### Example 3: Status RAM Access

```
; Set RAM bank 0, chip 2
LDM 0
DCL            ; Bank = 0
FIM 0P, 0x20   ; R0=2, R1=0
SRC 0P         ; chip=2

; Write to status register 2
LDM 15
WR2            ; Write 15 to status register 2 of chip 2

; Read from status register 2
RD2            ; Read status register 2 to ACC
```

## Example 4: Shift Register Chain

```
; Shift data into Chain A (ROM chip 15)
FIM 0P, 0xF0   ; R0=15, R1=0
SRC 0P          ; Select ROM chip 15

; Clock pulse with data bit = 1
LDM 3          ; bit0=1 (DATA), bit1=1 (CLK)
WRR             ; Rising edge on CLK: shifts 1 into 4003 #0

; Shift data into Chain B (RAM bank 3, chip 1)
LDM 1          ; Bank 1
DCL             ; Bank = low nibble of ACC (here: 1)
; ... need to set to bank 3
LDM 3
DCL             ; Bank = 3
FIM 0P, 0x10   ; R0=1, R1=0
SRC 0P          ; chip=1 (absolute chip 13 = 3*4 + 1)

LDM 3          ; bit0=1 (DATA), bit1=1 (CLK)
WMP             ; Rising edge on CLK: shifts 1 into 4003 #1
```

## Example 5: Multiple Peripherals on Same Port

**Setup:**

- ROM chip 0 port has three sources:
  - Internal chip value (from previous CPU write): 3 (0b0011)
  - 4-DIP switch (external peripheral): 5 (0b0101)
  - Push button (external peripheral): 2 (0b0010)

**CPU read:**

```
FIM 0P, 0x00   ; R0=0, R1=0
SRC 0P          ; Select ROM chip 0
RDR             ; Read port to ACC

; Result: ACC = 0b0011 OR 0b0101 OR 0b0010 = 0b0111 = 7
```

**Explanation:**

- The port value is the OR combination of all sources
- Bit 0: 1 (from any source)
- Bit 1: 1 (from any source)
- Bit 2: 1 (from DIP switch)
- Bit 3: 0 (no source drives it high)

## Important Notes

1. **ROM addressing**: 12-bit addresses access 4096 bytes total (16 chips × 256 bytes)
2. **RAM banks**: A full MCS-4 RAM stack is **16 chips** → DCL banks **0..3** only (4 chips per bank). Values 4..15 are not used to address more RAM in that standard layout.
3. **RAM chip absolute**: `chipAbs = (DCL_bank * 4) + chipRel`
4. **Shift registers**: Chain A from ROM chip 15 port, Chain B from RAM chip 13 port
5. **Port latches**: Each chip maintains its own port latch for edge detection and state retention
6. **OR-wired model**: Multiple peripherals on same port are OR-combined (simulates wired-OR bus)
7. **ROM port read**: `RDR` reads the OR-combined value (internal chip value OR all peripheral values)
8. **RAM port read**: Not directly accessible via CPU instruction (`RMP` is not a standard 4004 instruction)
9. **Status RAM**: Per-chip, 4 nibbles (0..3), accessed via `WR0..WR3`/`RD0..RD3`
10. **Snapshot frequency**: Separate from execution speed; controls UI update rate only

# Quick Reference

## Address Ranges

| Component | Range | Description |
|---|---|---|
| ROM Address | 0..4095 | 12-bit program address |
| ROM Chip | 0..15 | Chip selection |
| ROM Byte | 0..255 | Byte within chip |
| RAM Bank | 0..15 | DCL bank selector |
| RAM Chip Rel | 0..3 | Chip within bank |
| RAM Chip Abs | 0..15 | Absolute chip number |
| RAM Register | 0..3 | Register within chip |
| RAM Digit | 0..15 | Nibble within register |
| Status Index | 0..3 | Status register number |

## Instruction Summary

### Memory Access:

- `WRM` / `RDM`: Data RAM write/read
- `WRn` / `RDn`: Status RAM write/read (n = 0..3)
- `WMP`: RAM port write (supported)
- `WRR` / `RDR`: ROM port write/read
- `RMP`: RAM port read — **not** implemented in the CPU decoder; the bus can read RAM port latches for snapshots/debug only
- `WPM`: Decoded but **no-op** in this workbench (4008/4009 program-memory path out of scope)

### Control:

- `DCL`: Set RAM bank (from ACC)
- `SRC r`: Set RAM selection (chip/reg/digit) and ROM I/O chip
- `FIM r,#imm8`: Load 8-bit immediate into register pair
- `FIN r`: Load from ROM to register pair (via R0:R1 pointer)

### Shift Register:

- Clock via ROM chip 15 port (bit1 = CLK, bit0 = DATA) → Chain A
- Clock via RAM chip 13 port (bit1 = CLK, bit0 = DATA) → Chain B

# LEGAL NOTICE

> **Legal Notice and Fair Use Disclaimer:**

This project, Quadium 4004 Workbench, is an independent educational tool developed for historical preservation, research, and instructional purposes regarding early computing architectures.

**Trademark Acknowledgement:** Intel, MCS-4, 4001, 4002, 4003, and 4004 are registered trademarks of Intel Corporation. Texas Instruments, TMS9918, and related VDP marks may be trademarks of Texas Instruments Incorporated. These terms are used herein solely for nominative purposes to describe technical compatibility and historical context, which constitutes "fair use" under intellectual property laws. All references to these products are made for descriptive and compatibility purposes only.

**Non-Affiliation:** This project is not affiliated with, authorized, sponsored, or endorsed by Intel Corporation or Texas Instruments Incorporated.

**Original Documentation:** This documentation is an independent work created for Quadium 4004 Workbench. QuadBasic is an original language implementation designed for educational use with the emulated MCS-4 system.

**Third-party marks and materials:** Intel, MCS-4, 4004, 4001, 4002, and 4003 may be trademarks of Intel Corporation. Texas Instruments, TMS9918, and related VDP marks may be trademarks of Texas Instruments Incorporated. This product is not affiliated with or endorsed by Intel or Texas Instruments. Quadium does not distribute third-party ROM dumps, Intel manuals, Texas Instruments manuals, or datasheets. QuadBasic sample programs supplied with the product are authored by the publisher and transpile to 4004 assembly. Users are responsible for the rights to any files they load into the simulator.

**Nature of Simulation:** All logic described in this documentation represents an independent software implementation of publicly documented hardware specifications.

**Document version:** 0.80 (WIP)
**Date:** May 2026
**Language:** QuadBasic