

Micah Richards

Microcomputer Design Project

04/24/2018

# Description of Project

The microcomputer design project is a task requiring the production of a single board computer from scratch. The project necessitates the selecting of IC chips, designing of a printed circuit board, programming of chip communication procedures, and the programming of a monitor program for user interactions. At a minimum, the computer shall have an RS-232 compatible data port (Tx Data, Rx Data, CTS, RTS, Signal and Chassis Ground) capable of operating at 19,200 baud, an 8-bit bi-directional parallel data port with “handshaking” capability (8 data lines, ground, one control line to and one control line from the external circuitry) (optional), 64K of RAM (random access memory), the capability of addressing 64K of memory space, and an “on-board” monitor program residing in EEPROM. The monitor program shall be capable of examining the contents of any memory location within the memory space, changing the contents of any random-access memory location within the memory space, examining and changing the contents of any of the registers within the MPU, transmitting and receiving data via all data ports, allowing a machine-coded program to be loaded into RAM from a data port, and executing a program that has been loaded into RAM. This report details a design meeting these specifications.

# Hardware Description

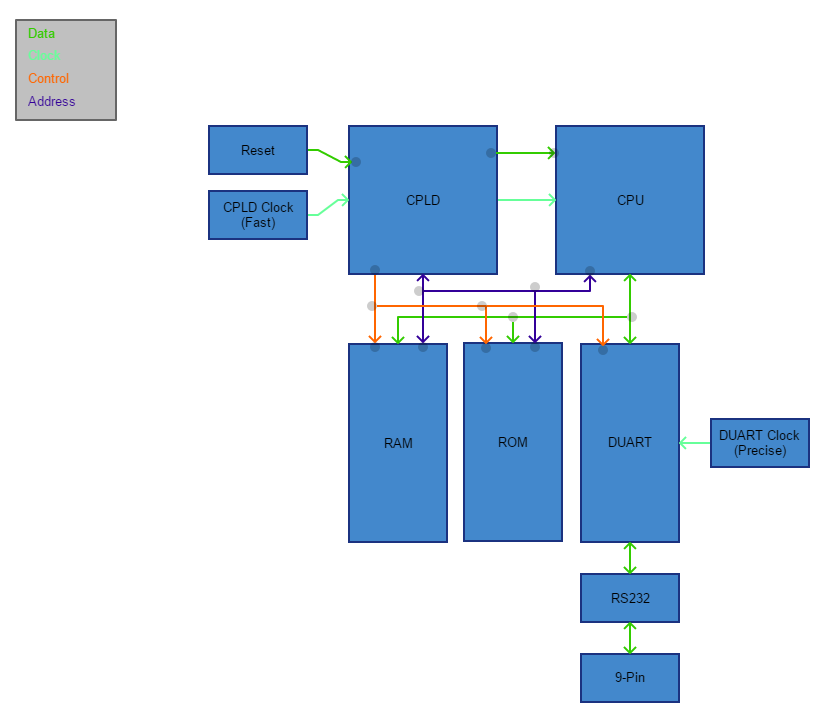
## General Introduction

The hardware is composed of 8 IC chips soldered to a custom printed circuit board manufactured by 4PCB. As errors were encountered, breadboards and perf-board based break out boards were used to provide space for necessary modifications.

## Device and Chip Selections

This project makes use of the Motorola 68000 CPU for its primary functionality. For memory, Atmel AT28C256 chips were selected for the ROM, while ZMD U62256A chips were selected for the RAM. Chip communication is made possible using a Xilinx XC9572 CPLD, and serial communications is enabled via a Philips SCC68681 DUART. The choice of these ICs was based on peer recommendations and past student’s recommendations for the given task.

## Hardware Block Diagram



*Figure 1 The initial conception of the linking between ICs*

# Software Description

## General Introduction

The software for this project is divided into two sub-components. The first is the chip selector logic, written in VHDL, allowing the CPU to communicate with the RAM, ROM, and DUART. The second portion is the monitor code, written in 68000 Assembly, allowing a user to interface with the micro-computer via another computer’s serial port.

## Memory Mapping

The chip selection portion of the software uses a simple flow of conditional statements to enable the proper chip at the proper time. Each cycle, every chip is deactivated; a comparison is then made of the CPU’s current outputs, and the proper chip is enabled. For this device, address lines 20-23 are tied to the CPLD for the purpose of selecting the device targeted by the CPU. In this design, ‘F’ signifies a selection of the RAM chips, ‘0’ signifies the selection of the ROM chips, and ‘3’ a selection of the DUART. Further differentiation on which chip needs selection is provided by testing the CPU’s Address High and Address Low pins.

## Detailed Description of Core Code

The monitor program is composed of seven unique features which can be selected by the user. At its beginning, the software resets all registers used in its operation. This is primarily to avoid mistakes in leftover data interfering with later operations. Following this, the menu is printed and the user is prompted for a selection.

### Read From ROM

The first menu option allows the reader to read 4 bytes from a specified memory address in ROM. The user is prompted for an address and the program displays the contents at the given memory address.

### Read From RAM

The second menu option functions in much the same way as the first, with the exception that the program appends a ‘0F’ to the address rather that ‘00’ to retrieve contents from RAM.

### Read From A Register

The third section of the program has a different operation than the first two, as it prompts the user for a register rather than a location in memory. This is done by copying a selected registers contents into the D0 register, printing the contents, and then restoring the CPU to its previous state.

### Write To RAM

The fourth option is the first to allow data to be written, enabling a user to modify data in the RAM. Like Read From RAM, the program prompts the user for an address and prepends the necessary ‘0F’ to the address. Following this, the program prompts the user for 4 bytes via 8 ascii characters ‘0’-‘F’ which are then written to the given location.

### Write To A Register

The fifth option allows the user to modify a register’s contents. Note, some registers will not appear to change due to their reset at the beginning of the menu printout. Additionally, in some situations, more than just the selected register will be modified to ensure the stack is restored properly at the end of the procedure. The program will prompt the user for a register from ‘A0’-‘A7’ or ‘D0’-‘D7’, then request 8 ascii characters for the new memory contents.

### Write S-Record

The sixths option allows the user to load an S-Record into memory. This device accepts only S2 and S8 records, with any other type causing an error and returning the user to the menu. Invalid data will also cause an error. The program will prompt for the data to be entered, and will exit after the S8 record concludes the upload.

### Run S-Record

The seventh option jumps the program to the beginning of the S-Record program. The uploaded program will be executed, and the user will then be returned to the main menu.

## Flow Chart

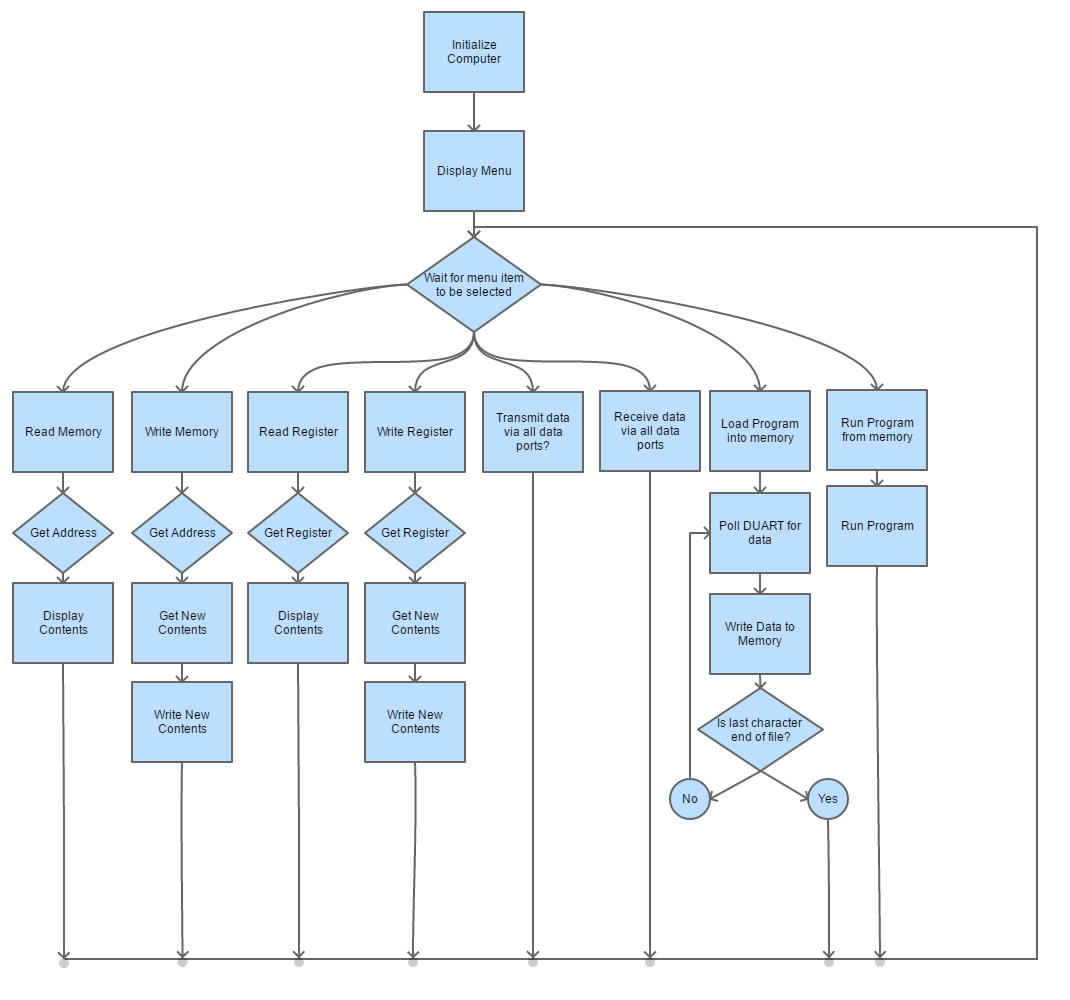


Figure 2 The Initial Concept For The Flow Of The Monitor Program

# Troubleshooting

## Hardware

### Socket Sizes

The first issue discovered while making this project was the realization that the CPLD place on the PCB had been designed for a 68-pin IC rather than the 84-pin IC that had been purchased. Additionally, the RAM chips were too wide for their position on the board. To solve this problem, perf-board was used to create adapters between the PCB and the sockets, using solder bridges to connect the pins on the socket with header pins spread out to the necessary locations.

### Pin-Layout Error

The CPLD required further work as there is no way to adapt 84 pins to fit into 68 holes, thus braided copper wire was purchased with female headers which could be run from the CPLD breakout board to the PCB in any configuration needed. This method also allowed for changes to be made as later issues were discovered, without the need to resolder connections.

### Poor CPU Socket

The CPU socket purchased did not hold the CPU well enough to ensure stable connection. The connection was so poor, the CPU would dislodge from its position after nearly every test. This was resolved by soldering the chip directly to the socket.

### Missing Clock Trace

The CPU Clock trace was missing its final segment, whether due to damage during soldering or a flaw in the board during manufacturing. The schematic shows the trace connected across the missing section. This was resolved by applying a solder bridge across the missing trace.

### Damaged CPLD Clock

While the CPLD Clock seemed to be functioning well according to the logic analyzer, a test with the oscilloscope showed it alternated between 2.2V and 2.7V rather than 0V and 5V. Replacing this clock showed that the spare was also malfunctioning. This was ultimately solved by running the CPLD off the clock intended for the DUART.

### Electromagnetic Interference From Clocks

After changing the clocks, there was still periodic interruptions in the monitor program. This was decided to be due to the clocks and their proximity to power lines. This was resolved by moving the clocks to an external breadboard.

### Poor Power Lines

Due to continuing interruptions in the monitor program, it was decided there was not enough power reaching the CPLD. This was resolved by changing from braided wire to solid core.

### Failing Pull-Up Of DUART DTACK

The DTACK pin of the DUART, when probed, showed it was failing to go high when the DUART did not have data with which to interact. This was tied high with a 1kΩ resistor to solve the problem.

### Poor Socket Connection To DUART

Constant probing of the DUART damaged the pins in the socket. This was resolved by inserting bare wires between the damaged pins and the DUART to reconnect the pins.

## Software

### Write To RAM Failure

When the monitor program was tested, the Write To RAM feature was continually failing. The error was due to a register being restored at the end of a sub-procedure and overwriting the calculated data. A change of this code resolved the issue.

# Conclusion

While there were many hours of debugging necessary to bring this project to the desired level of functionality, it was a useful project for understanding the intricacies necessitated by computers. I have a deeper appreciation for the complexity of modern devices and a better understanding of how they function. This knowledge will be useful in my work as a software engineer as I now better understand the methods used to enable the higher-level code I write. Further, I have learned many useful techniques in debugging a system without direct visual information; specifically, the use of multimeters, digital logic analyzers, and oscilloscopes, each as required by the current issue.