8-Bit Microcontroller with Flash

Application Note

Using a Personal Computer to Program the AT89C51/C52/LV51/LV52/C1051/C2051

Introduction

This application note describes a personal computer-based programmer for the AT89C51/C52/LV51/LV52/C1051/C2051 Flash-based Microcontrollers. The programmer supports all flash memory microcontroller functions, including code read, code write, chip erase, signature read, and lock bit write. When used with the AT89C51/C52/ LV51/LV52, code write, chip erase, and lock bit write may be performed at either five or twelve volts, as required by the device.

Devices sporting a “-5” suffix are intended for operation at five volts, while devices lacking the suffix operate at the standard twelve volts.

The programmer connects to an IBM PC-compatible host computer through one of the host's parallel ports. Required operating voltages are produced by an integral power supply and external, wall-mounted transformer.

Software

Software for the programmer is available by downloading it from the Atmel BBS at 408-436-4309.

The programmer is controlled by software running on the host. The AT89C51/C52 and C1051/C2051 have dedicated control programs, which were written in Microsoft C. Programs dedicated to the AT89LV51/LV52 do not exist; these devices are supported by the programs for the AT89C51/C52, respectively. In the text below, all references to the AT89C51/C52 may be assumed to apply to the AT89LV51/LV52 as well.

All programmer control programs are invoked from the DOS command line by entering the program name followed by “LPT1” or “LPT2” to specify parallel port one or two, respectively. If the parallel port is not specified, the program will respond with an error message. The control programs are menu-driven, and provide the following functions:

Chip Erase

Clear code memory to all ones. The successful operation of this function is not automatically verified.

Program from File

Write the contents of the specified file into device memory. The user is prompted for the file name, which may require path and extension.

The file is expected to contain binary data; hex files are not accepted. The first byte in the file is programmed into the first location in the device. Successive bytes are programmed into successive locations until the last location in the device has been programmed or until the data in the file has been exhausted.

Programming occurs regardless of the existing contents of device memory; a blank check is not automatically performed. After programming, the contents of device memory are not automatically verified against the file data.

Each programmed location in the device receives the maximum programming time specified in the data sheet. This is done because timing is enforced by software; the programming status information provided by $\overline{\text{DATA}}$ polling and RDY/$\overline{\text{BSY}}$ is not utilized.

The control program provides no visual indication that programming is in progress. The main menu is redisplayed when programming is complete.

0285D-B–12/97

Verify against File
Compare the contents of code memory against the contents of the specified file. The user is prompted for the file name, which may require path and extension.

The file is expected to contain binary data; hex files are not accepted. The first byte in the file is compared to the first location in the device. Successive bytes are compared to successive locations until the last location in the device has been compared or until the data in the file has been exhausted.

Locations which fail to compare are displayed by address, with the expected and actual byte contents. If there are no compare failures, nothing is displayed.

Save to File
Copy the contents of device memory to the specified file. The user is prompted for the file name, which may require path and extension. The number of bytes in the resulting file is the same as the number of memory locations in the device.

Blank Check
Verify that the contents of device memory are all ones. Only pass or fail is reported; the addresses and contents of failing locations are not displayed.

Read Signature
Read and display the contents of the signature bytes. The number of signature bytes and their expected contents varies between devices. Refer to the device data sheet for additional information.

Write Lock Bit 1
Write Lock Bit 2
Write Lock Bit 3
Set the indicated lock bit. Note that the AT89C1051/C2051 contain only two lock bits, while the AT89C51/LV51 and AT89C52/LV52 contain three lock bits. The state of the lock bits cannot be verified by direct observation.

Exit
Quit the programmer control program.

System Dependency

The control programs for the AT89C51 and AT89C52 come in two flavors: host system-dependent and host system-independent. System-dependency results from the use of software timing loops to enforce required delays, the duration of which will vary between host systems running at different speeds. The code provided was tested on an 80386-based system running at 33 MHz, and may require modification for use on other systems. This method was chosen for its simplicity.

Host system-independence is achieved by using the Programmable Interval Timer embedded in the system hardware to enforce time delays independent of system speed. The timer is reconfigured when the control program is invoked and restored to its original state before the program terminates. In order to guarantee that the program is not exited before the timer configuration is restored, the CTRL-C and CTRL-BREAK keys are disabled. This means that the program cannot be aborted except by specifying the exit option at the main menu or by rebooting the system.

The timer control code is provided as an 8086 assembly language module, which is linked with the compiled control program. The granularity of the timer is 0.838 microseconds, but the minimum practical delay is system- and software-dependent. The timer code ensures that the delay produced will not be of shorter duration than requested.

The control programs provided for the AT89C1051/C2051 are system independent.

Programmer

The programmer circuitry (see Figures 1 and 2) consists of the host interface and switchable power supplies. The signal sequencing and timing required for programming is generated by the host under software control. A 40-pin ZIF socket is provided for programming the AT89C51/C52; the 20-pin ZIF socket accommodates the AT89C1051/C2051. Note that the power and ground connections and bypass capacitors required by the TTL devices are not shown on the schematic.

Power for the programmer circuitry and the AT89C51/C52/C1051/C2051 is provided by a fixed five volt supply. A second supply provides either five or twelve volts, selectable, for use during programming. The addition of a transistor to the output of the variable supply provides a third level, ground, for use when programming the AT89C1051/C2051.

The resistor values utilized in the variable power supply circuit were determined using the equations presented in the LM317 voltage regulator data sheet. Power supply ramp rates are accommodated by the host software. For 5 V-VPP programming, the devices must be ordered from the factory as an AT89CX-XX-5 (not available with the AT89C1051/2051).

The programmer is connected to the host with a 25-conductor ribbon cable. To minimize the effect on signal integrity, the length of the cable should be as short as possible, preferably not exceeding three feet.

Parallel Interface

The original parallel interface provided by IBM was probably not intended to support bidirectional data transfers. However, due to the way in which the interface was implemented, bidirectional transfers are possible. Over the years, many products have appeared which exploit this capability.

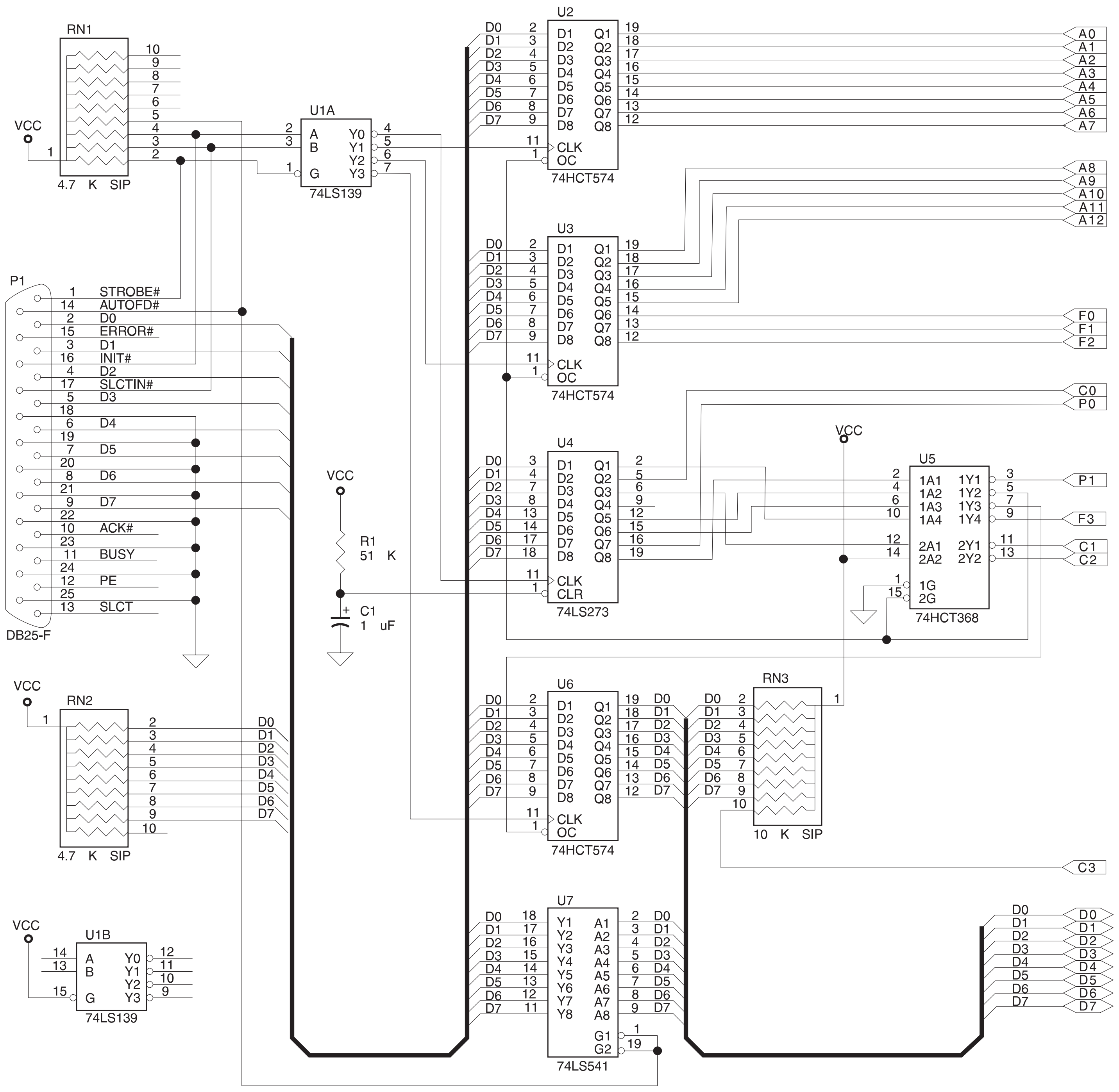
Unfortunately, many system and interface card manufacturers have not faithfully cloned the IBM design, resulting in bus contention when the peripheral attempts to drive return data into the interface. Usually the peripheral drivers can overpower the interface drivers and the peripheral works, though this is not considered a good design practice.

Most parallel interfaces are now implemented in a single chip, such as the 82C411 or 16C452. These chips allow their output drivers to be disabled under software control, providing true bidirectional operation. The programmer software automatically enables bidirectional operation when used with parallel interfaces utilizing the 82C411, 16C452, or similar chips.

Note that these chips also possess a mode control pin which must be at the correct level to enable the directional control feature. As a result, parallel interfaces utilizing these chips cannot be assumed to be bidirectional.

If the programmer writes devices, but fails to verify, or the signal levels at the interface don't meet TTL specifications, the parallel interface may be incompatible with the programmer. A design is provided (see Figure 4 and Figure 5) for a parallel interface which supports bidirectional operation and is compatible with the programmer. This design is simple, requiring only six ICs. The interface can be strapped to appear as LPT1 (addresses 378-37F hex) or LPT2 (278-27F hex) and will be recognized by the POST when the host system is powered up. Due to its simplicity, the parallel interface CANNOT be used as a printer interface.

Figure 1. AT89 Series Programmer Interface



Note: 0.1 µF bypass caps on all ICs

Figure 2. Power Supply for AT89 Series Programmer

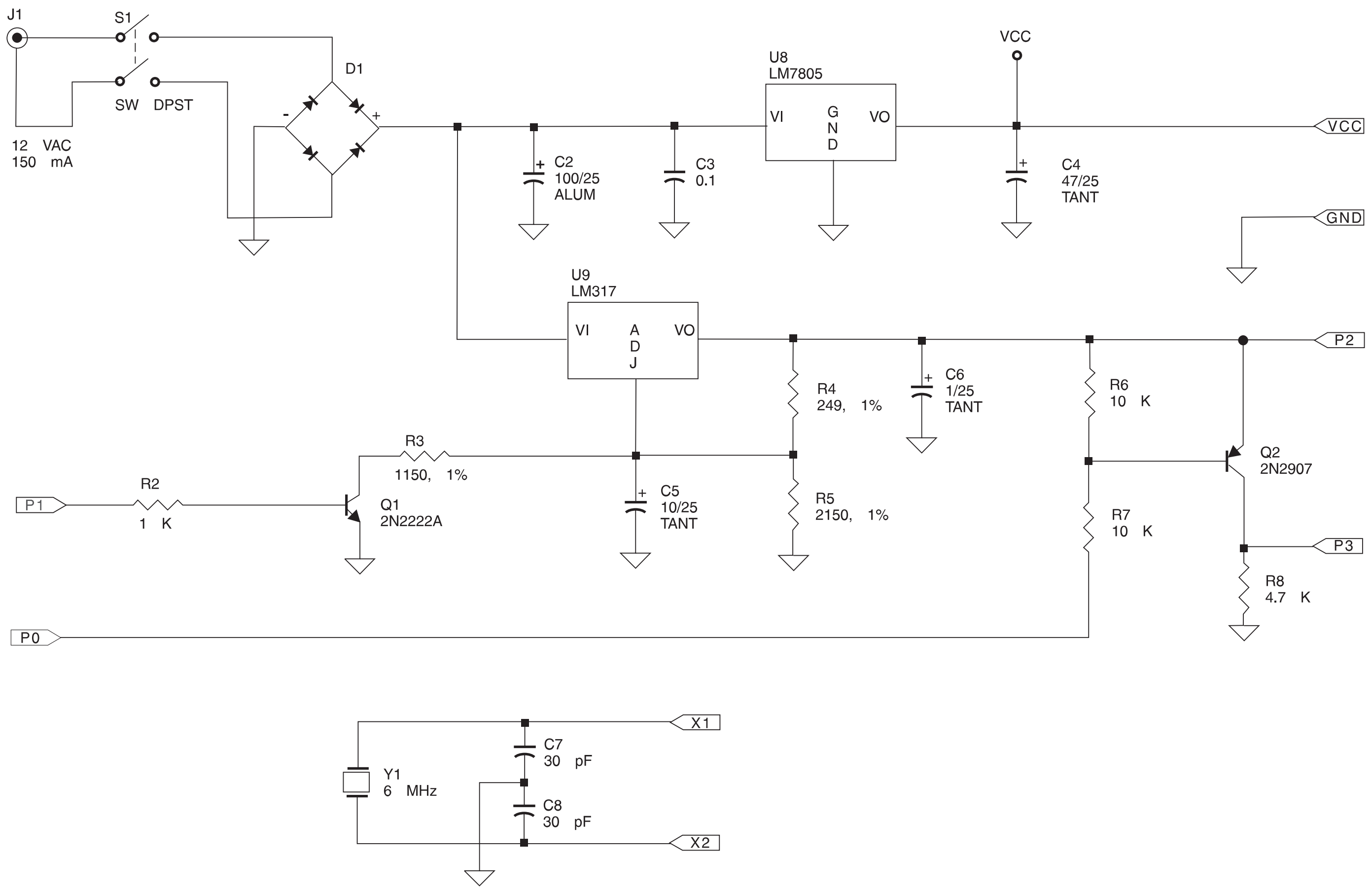


Figure 3. AT89 Series Programmer Socket Wiring

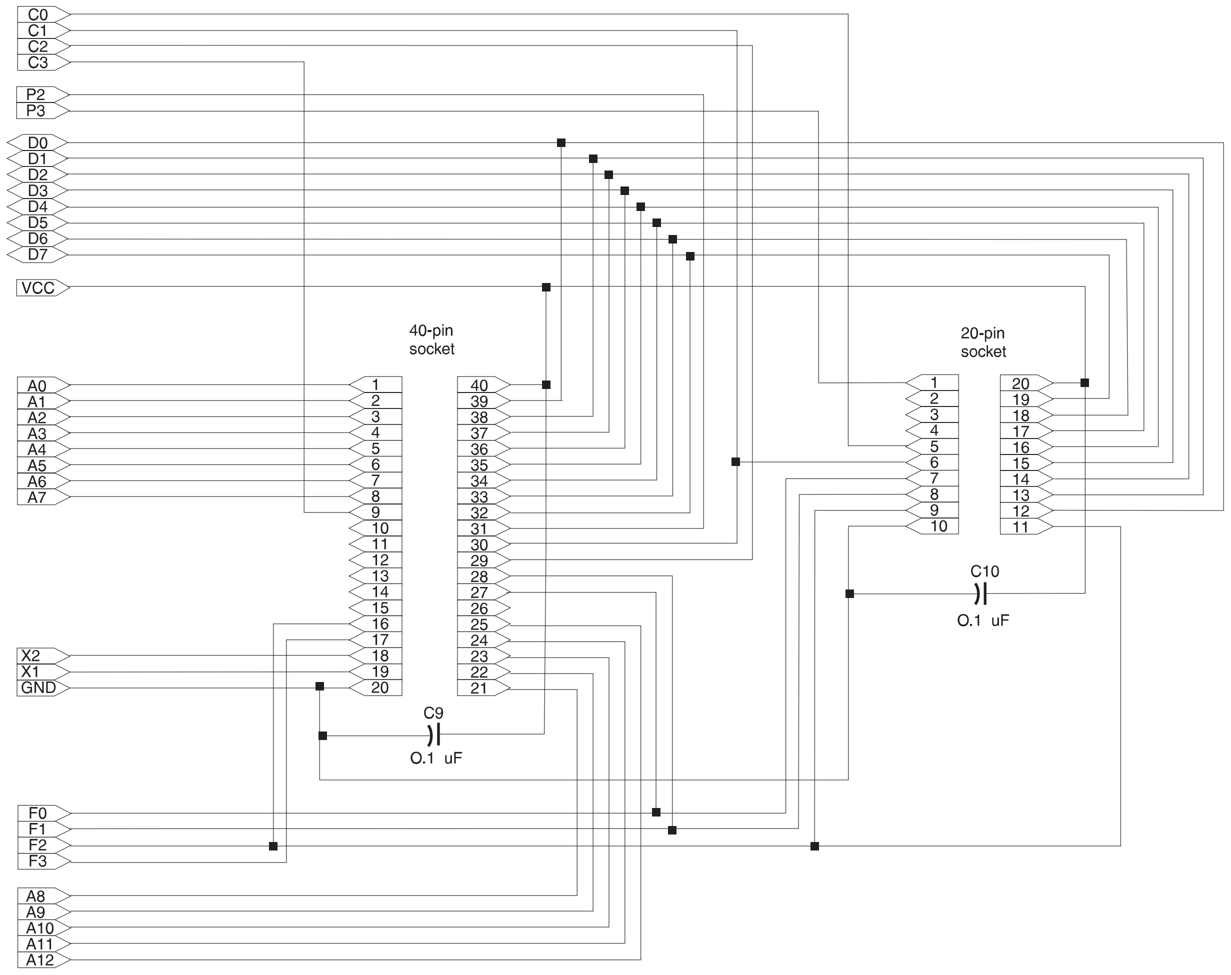


Figure 4. A Parallel Interface Supporting Bidirectional Operation

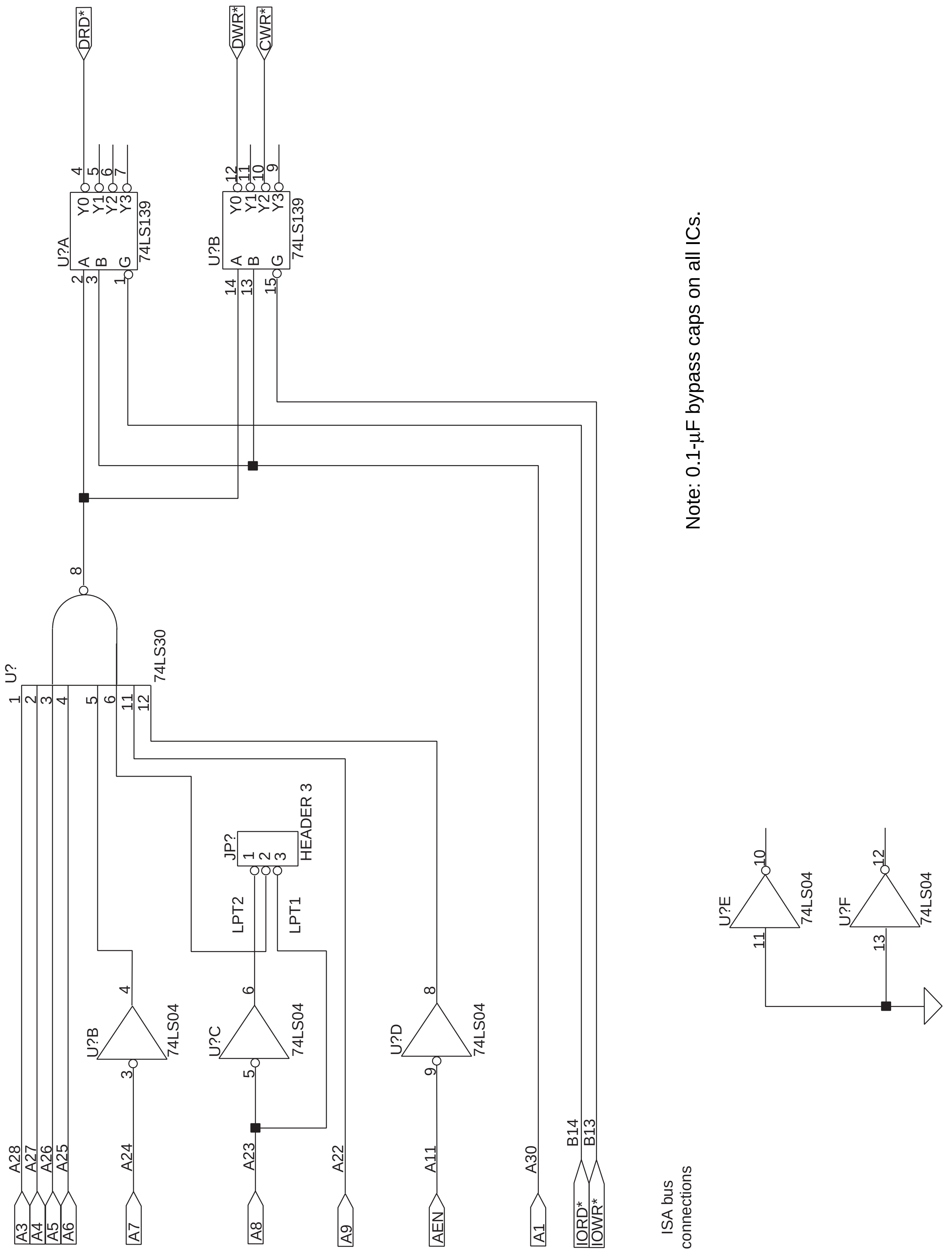
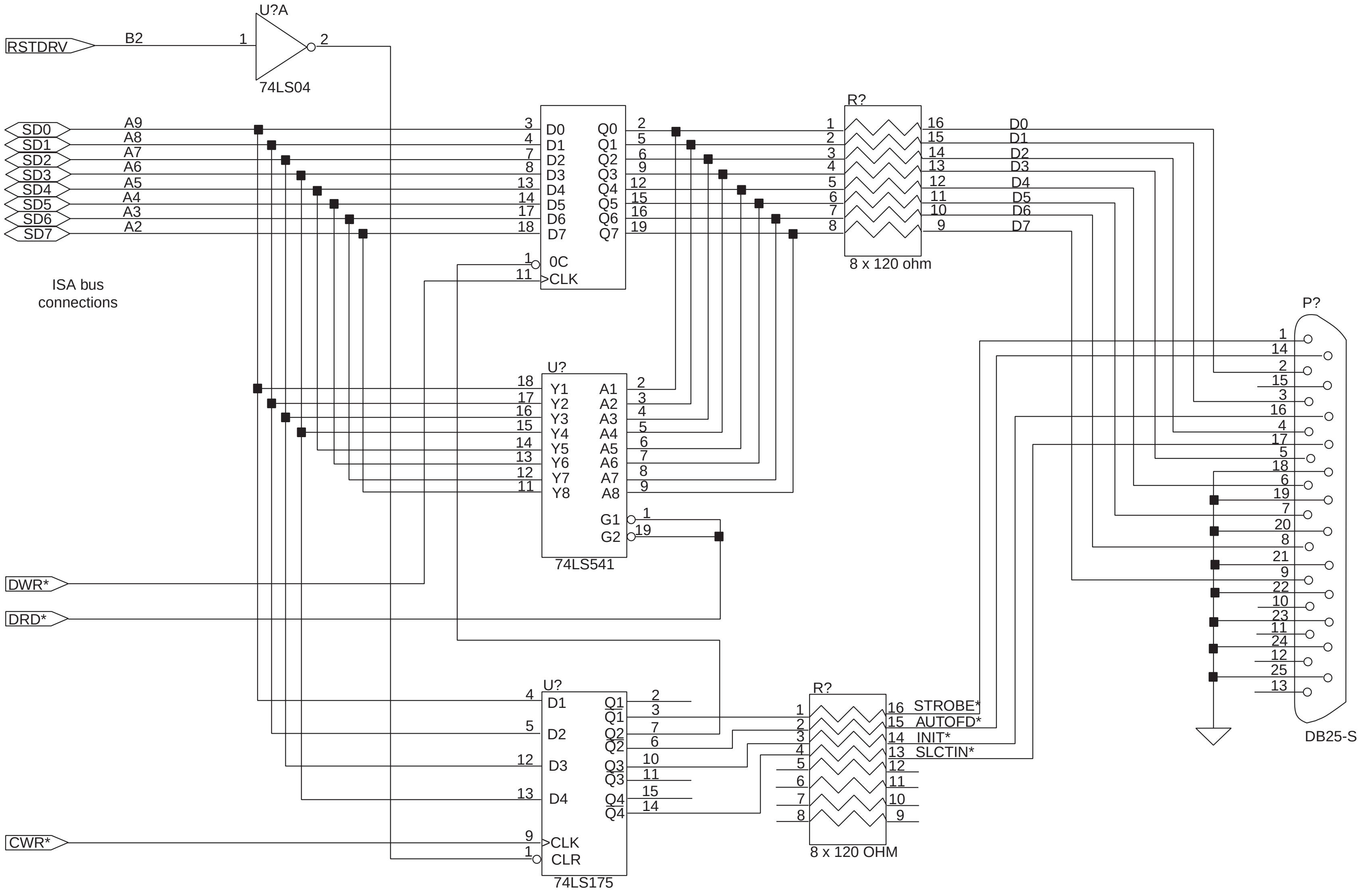


Figure 5.



Note: 0.1-µF bypass caps on all ICs.