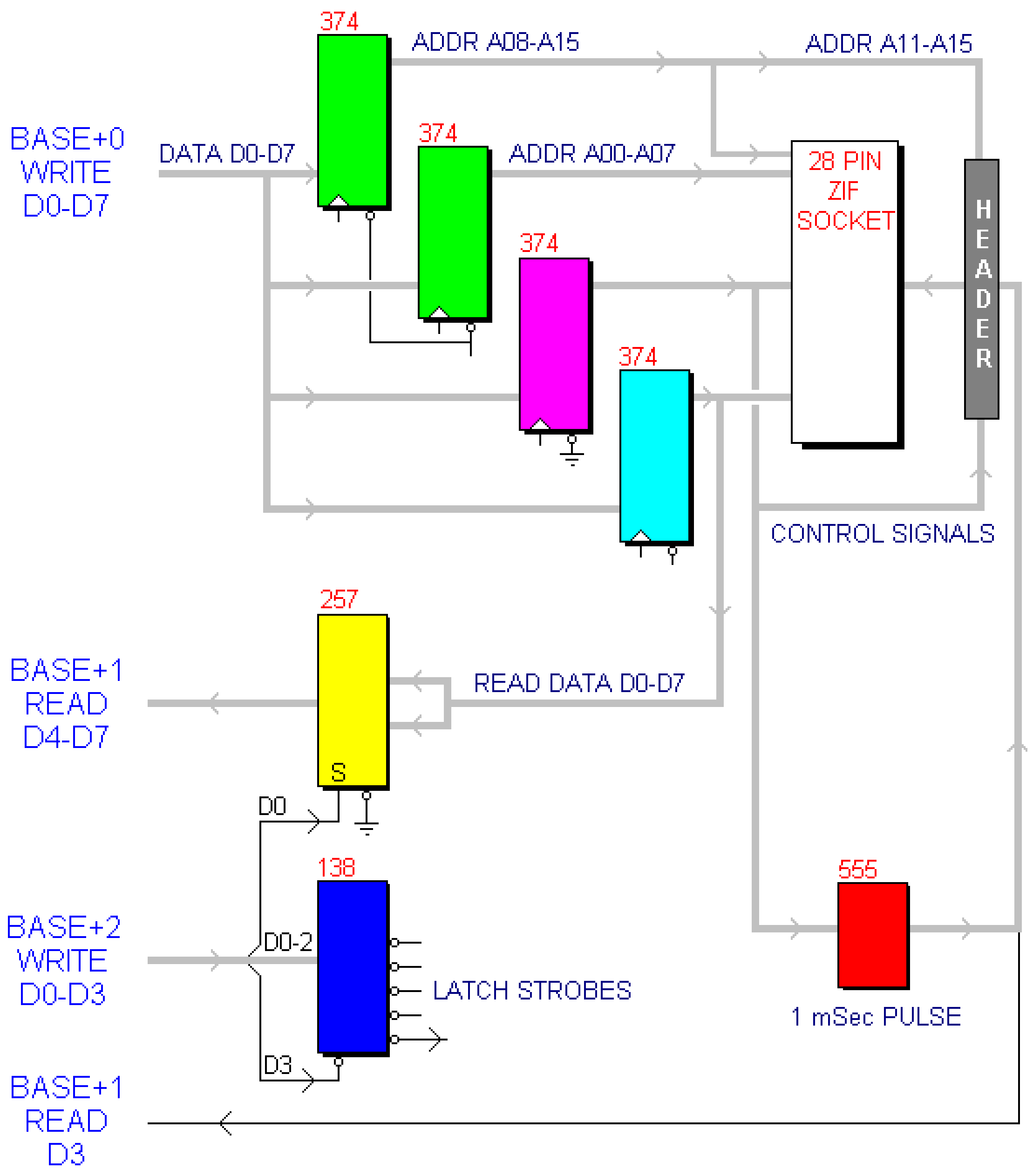
SIMPLE EPROM PROGRAMMER

A simple EPROM programmer is described here. It connects to a PC's parallel port and can program EPROMs from 2716 to 27512. The design can be expanded to program even larger EPROMs. DOS based software is available to program EPROMS from various data file formats.

EPROM Programmer Design

This programmer is shown in outline form in the figure below. The basic philosophy is that the design uses common **TTL** ICs that should be readily available to all constructors. It is designed to connect to a standard (IBM **PC** Clone) parallel port and should work independently of the processor's speed.



Hardware

Four 74HC374 latches are used to latch 16 bits of address **(green)**, 8 data bits **(cyan)** and 8 control bits **(magenta)**.

Data for the four latches comes from the parallel port's 8 data lines D0-D7 (DB25 pins 2 through 9). Strobe signals to latch each of the 374s are generated by a 74HC138 decoder **(dark blue)**. This decoder is driven by the four parallel port **control lines** STROBE*, AUTO FEED*, INITIALISE, and SELECT IN* (DB25 pins 1, 14, 16, 17 respectively). Data is read back from the EPROM in lots of 4 bits on the **status lines** SELECT, PAPER END, ACK*, and BUSY (DB25 pins 13, 12, 10, 11 respectively). A 74HC257 multiplexor **(yellow)**, controlled by STROBE*, is used to select either the high or low order 4 data bits to be read back by the PC. A 555 timer **(red)** generates a 1 millisecond pulse that is supplied to the EPROM.

A header is used to customise the programmer for each EPROM type that may be programmed. Address lines A11, A13, A14, and A15, and other control signals generated by the programmer go to the header. Seven EPROM lines (pins 1, 20, 22, 23, 26, 27 and 28) come from the header.

Note that both the data and the two address latches can be disabled under software control. The data latch is disabled in order to read and verify the EPROM's data but is enabled during programming. The control latch is always enabled.

Software

Control over the programmer is achieved by writing to, and reading from, a standard parallel port that the programmer hardware is connected to. There are only three registers available on a standard parallel port:

- an 8 bit output data register at BASE+0,
- a five bit input status register at BASE+1,
- and a 4 bit output control register at BASE+2.

The value of BASE is the hardware address assigned to the port.

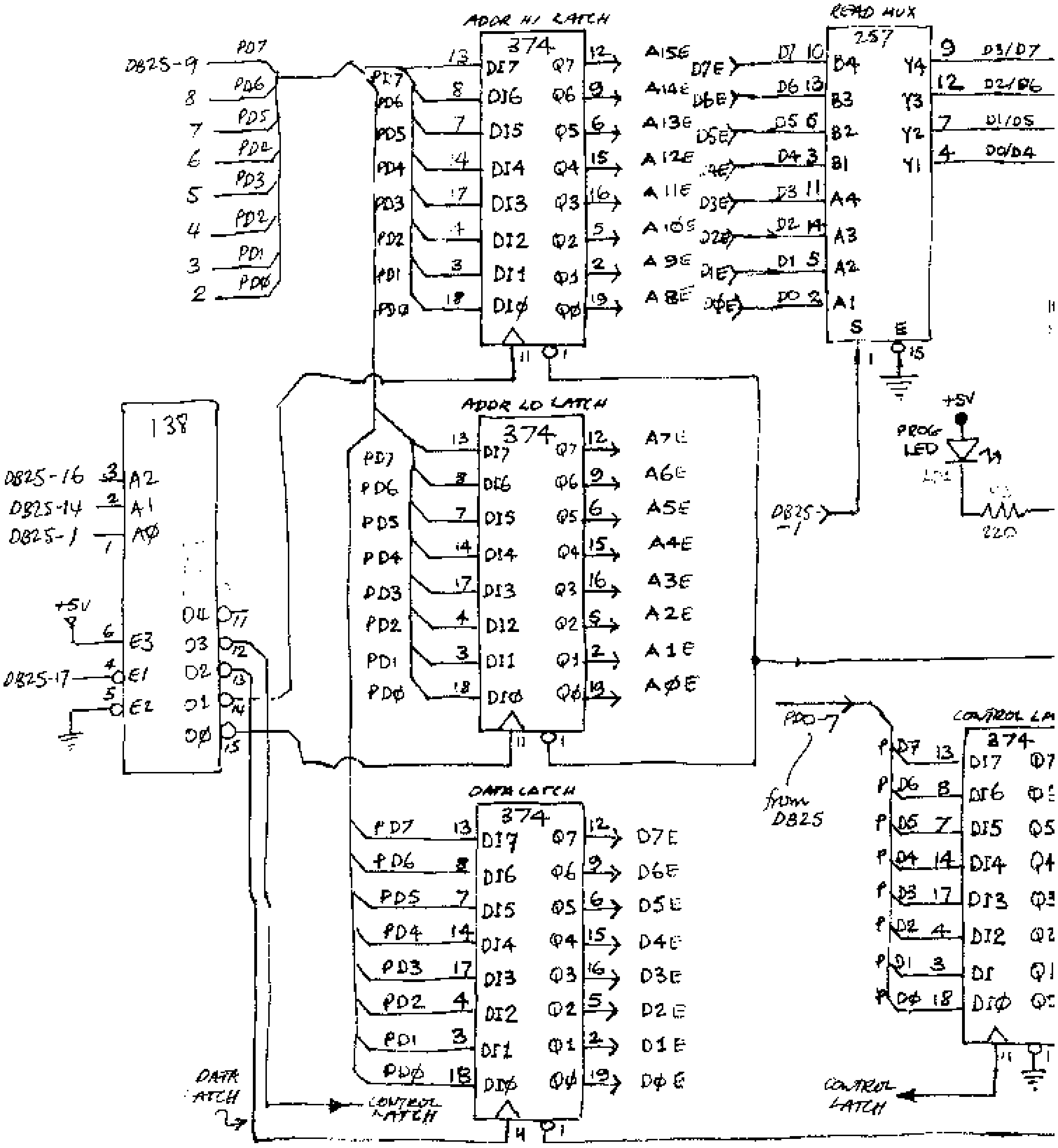
Any one specific latch on the programmer is written to using the following procedure. First the 138 is setup to select the specific latch (output of 138 is active low). This is done by writing to the parallel port's BASE+2 register and this produces a negative edge to the latch's trigger input and is ignored by the 374. Next the latch's input data is setup by writing to the port's data at BASE+0. Lastly the 138 is written to in order to deselect the latch, and this produces the positive edge which actually latches in the data to the 374.

To program a location in an EPROM the address, as a low and high byte, is first set up in the address latches. The data is then set up in the data latch. Finally the control signals are manipulated to generate a programming pulse. The programming pulse is generated by the 555 timer and this has a duration of 1 mSec. For EPROMS that require pulses longer than 1 mSec (many early EPROMs require 50 mSec pulses) the software must produce multiple pulses. For durations shorter than 1 mSec it is possible for the software to terminate the programming pulse.

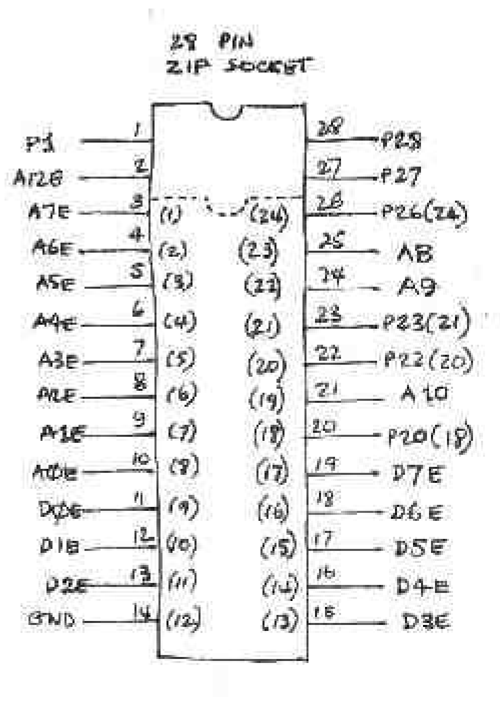
Programming verification requires that the data latch is tri-stated (the data 374's output is disabled) and the EPROM's data may then be read through the 257 multiplexor (mux) in two nybles (groups of 4 bits).

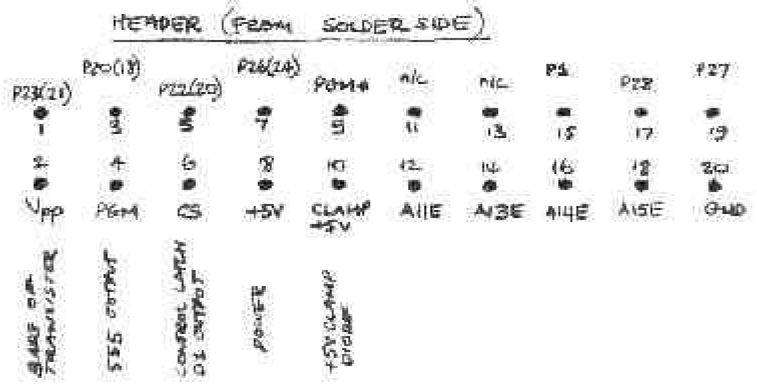
Programmer Schematic

The schematic for the author's prototype is as follows:

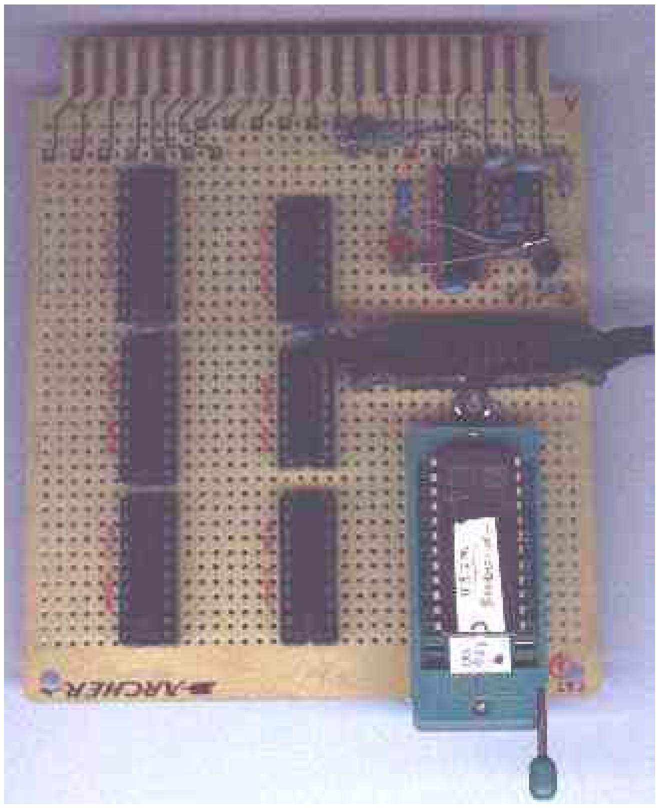


Here's the EPROM socket and header pinouts (individual headers not shown).





And here's a photo of the prototype (!).



We don't have a PCB, so, if you're interested in this project, and would buy a PCB from us, then contact us and we may have one soon.

Anyway, if you've read this, tell us. However, don't bother telling us the information is incomplete, or that we haven't made the software or a PCB design available (we know this!). Our final design will also probably have additional parallel port buffering and noise reduction circuitry to ensure reliable operation with long cables (that people insist on using!). This final point is the main reason we have not produced a final design yet.

We also have a design for a 2 chip mini-reader (working), and a 3 chip programmer (conceptial stage). Details soon...

To ensure the correctness of this document, we greatly appreciated your feedback on the information presented here.

Last updated 14 February 2001 (URLs).

Home | Feedback | Copyright