Core Loading Automaton

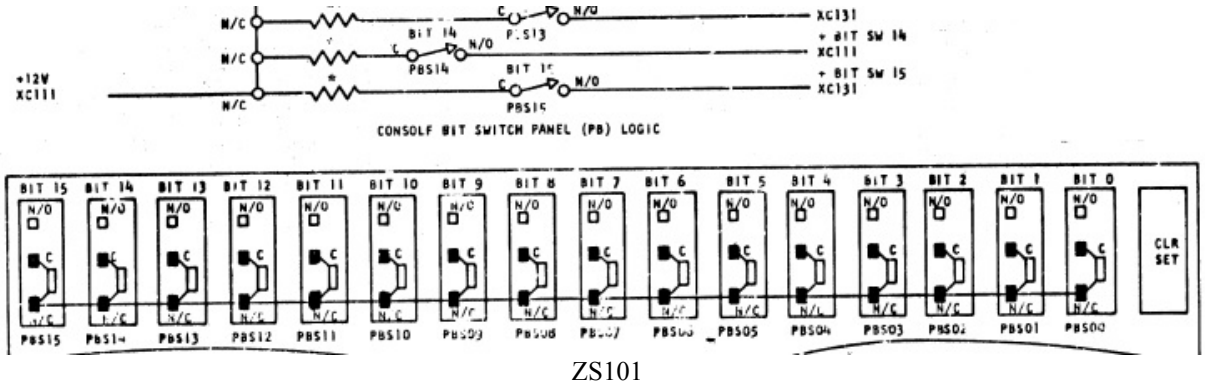
A tool developed at the request of Peter Vaughan of The National Museum of Computing, this small modification to an IBM 1130 will allow a PC based file of memory words to be loaded automatically into the core of an 1130.

The device operates by overriding the 16 console entry switches (CES) on the faceplate of the 1053 console printer and by simulating presses of the Program Start button and Load IAR button on the 1130 console.

When the machine is set to LOAD mode and has an appropriate starting core address set via the Load IAR button, it will enter a word from the CES into ascending addresses on each push of Program Start.

When an address (hex word prefixed by @) is received, it sets up that address in the CES and presses the Load IAR switch virtually. When it receives a hex word without the prefix, it enters it in the CES and pushes the Program Start button.

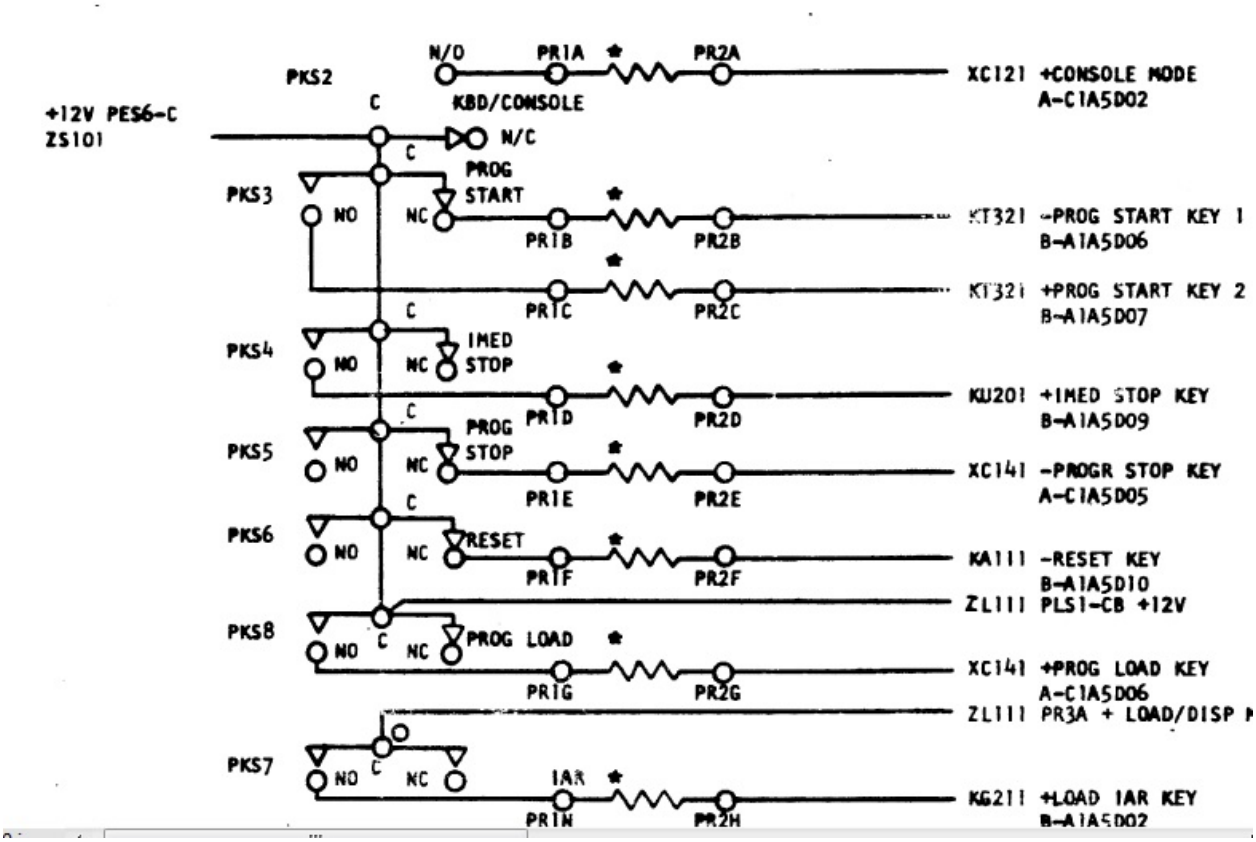
It communicates over a serial link on a USB cable, interpreting each line of text sent to it as four hex characters with some simple error checking. The line is converted to the 16 values for the CES. The device activates relays to power CES that have a 1 value, while another relay will virtually press the Program Start button.



ZS101

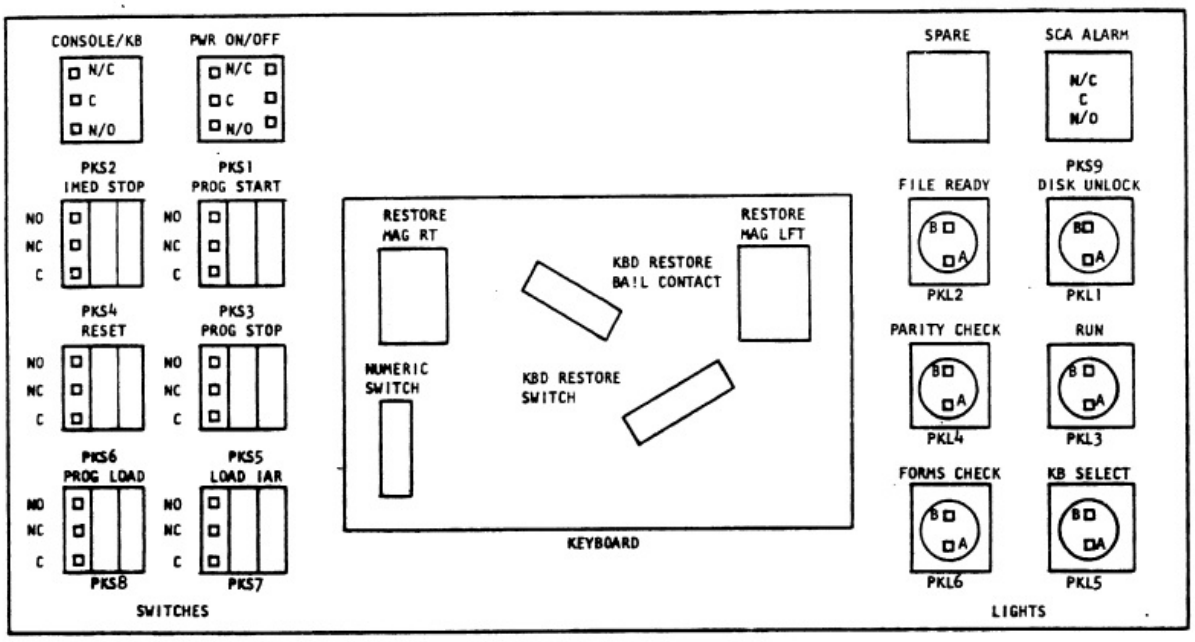
To install the device, we first have to add wires to the Normally Open contacts of each of the 16 CES switches (see ALD ZS101), each hooked to a small relay on a board that is controlled by an Arduino. Boards are readily found that house eight relays apiece, thus two boards will serve to drive all 16 CES and a 4 relay smaller board completes the unit.

We disconnect a wire on page ZK111 of the 1130 ALDs that hooks to the Common terminal of switch PKS3 (Program Start) - this is a run of wire carrying +12V to the common terminals of switches PKS2, PKS3, PKS4, PKS5, PKS6 and PKS8, but we break the connection to PKS3.



We tap on the +12V line that we just removed and run it to the common terminals of relays 1 to 18 of our device. The normally closed contact of relay 17 will be connected via a new wire to the common terminal of PKS3. Thus, in its normal deenergized state, the +12V is delivered to the Program Start (PKS3) switch. By activating relay 17, we disable the physical Program Start switch.

We run two new wires, one each from the NC and NO contacts of PKS3 back to the NC and NO contacts of relay 18. This will act as our virtual Program Start button. Another two wires, from PKS7 NO and C contacts, go to the NO and C contacts of relay 19.



The diagram above shows the underside of the console panel, with our PKS3 Program Start switch clearly shown along with the terminals to which we will connect wires (NO, NC and C) after removing the +12V power wire originally hooked to terminal C.

The three relay boards (two 8-relay and one 4-relay board) have their trigger connections wired to the Arduino, which can flip them on and off by program to activate the virtual Prog Start button, Load IAR button and the 16 CES switches.

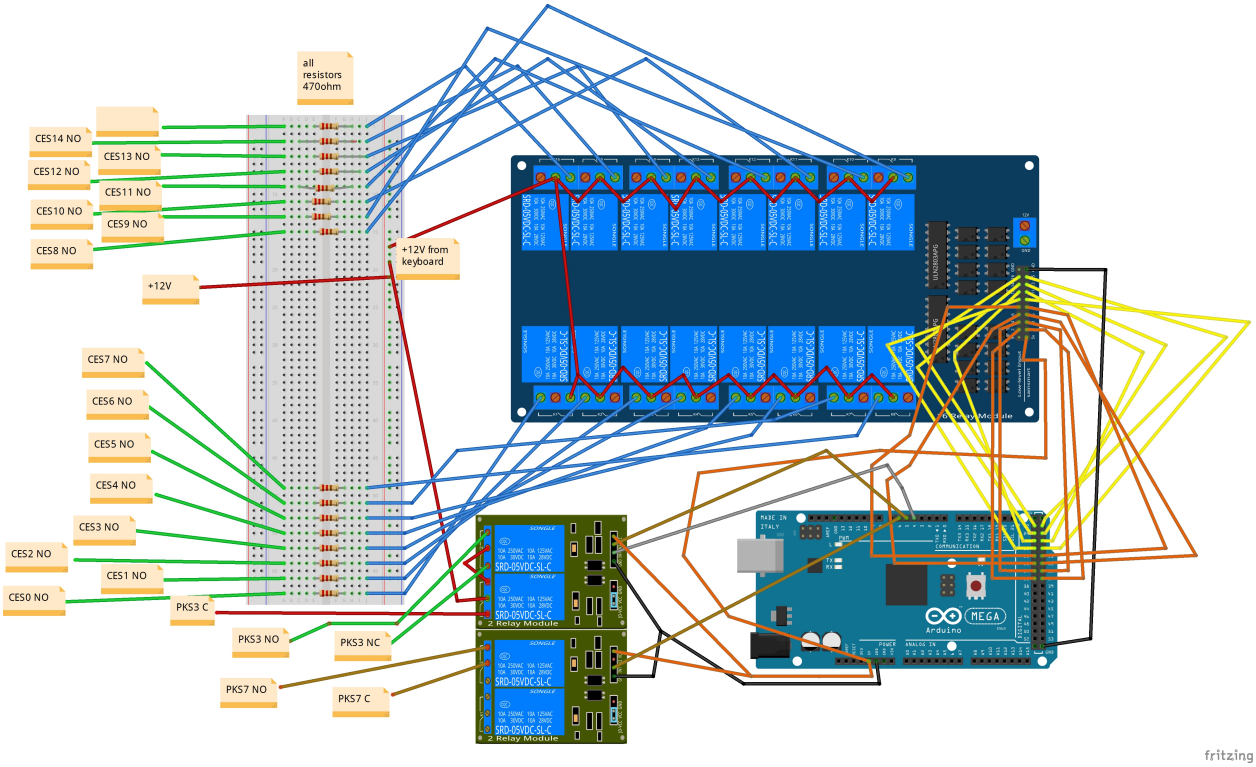
I will be drawing the wiring diagram based on relay boards similar to https://www.amazon.com/Xiuxin-Channel-Relay-Module-Arduino/dp/B07C8LSXKC/ref=sr_1_1_sspa?ie=UTF8&qid=1523998210&sr=8-1-spons&keywords=8+relay+board&psc=1
and https://www.amazon.com/SainSmart-101-70-101-4-Channel-Relay-Module/dp/B0057OC5O8/ref=sr_1_4?ie=UTF8&qid=1524107129&sr=8-4&keywords=4-relay+board

Any compatible alternative is fine, as long as it can be driven safely by an Arduino and provides at least a SPDT function for each relay.

Each relay has a three screw connection for its output, the center screw is the common pole. The normally closed contact is shown with a line drawn to the common screw, the third screw is normally open.

The inputs are on the small connector, each wired to an Arduino pin.

As noted by Peter, external power should be supplied since the demands of all the relays will exceed the ability of the Arduino to deliver USB based power to everything.



Wiring diagram (Fritz diagram of connections)

Software

At startup, the program activates the relay that disables the Program Start physical switch and routes the +12V to our virtual button relay. Thus, the USB cable should only be plugged into a PC when the loading function is desired since it will disable the real Program Start button.

The Arduino program loops listening to the serial port (USB link to PC) at 9600-N-8-1 setting. It builds up a line of four characters, if longer it is an error and if shorter the input is ignored. The code skips any NL (x0A) or CR (x0D). It looks for exactly four characters that are valid hexadeci-mal - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, or F. The letters can be upper or lower case ASCII.

Once it has a good word, it triggers the relays for the Console Entry Switches for each switch whose value is 1 from the input we just recei ved. For example, if we recieved the word 3F29 then we want to activate the relays for switchs 13, 12, 11, 10, 9, 8, 5, 3, and 0 which corresponds to 0x3F29.

After waiting a fraction of a second for the input to the switches to stabilise, we activate the relay for the virtual Prog Start button, leaving it pressed for half a second and then releasing it. Once it is released, we go back to give a prompt and accept the next word of entry from the terminal hooked to our USB cable.

If the user enters an @ in the line with the four hex characters, it sets the value on the Console Entry Switches and pushes the virtual Load IAR button, with similar timing to how it handles Program Start above.

This code is written for an Arduino Mega 2560 board. We need 19 output pins which excludes most of the smaller Arduino boards.

The code will be placed on Github as well as shared with any 1130 restorer wishing to use or modify this device.

The code was modified to fix a bug in the activate/deactivate logic and to provide a prompt character, supporting pacing of a terminal emulator or other serial communications software, otherwise the small buffers in the Arduino can be overrun.