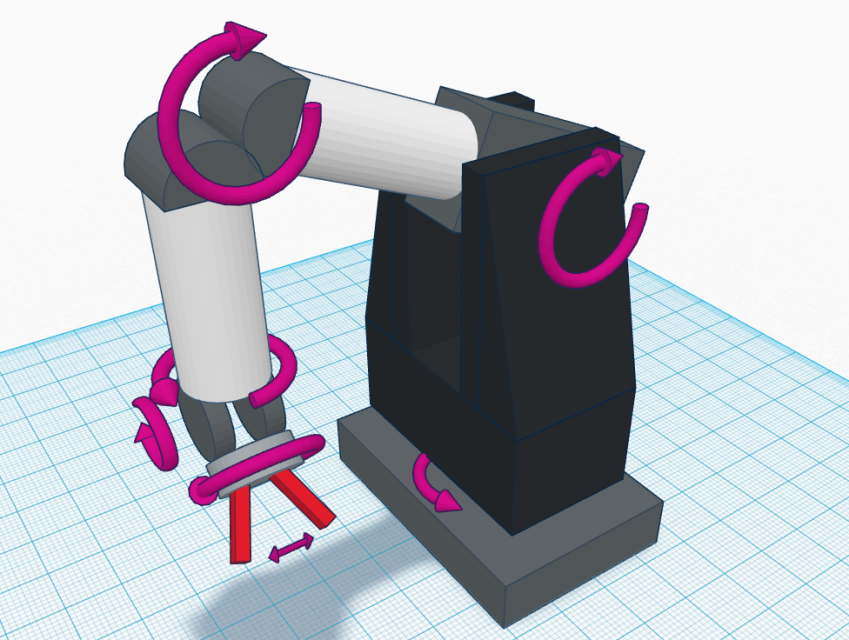
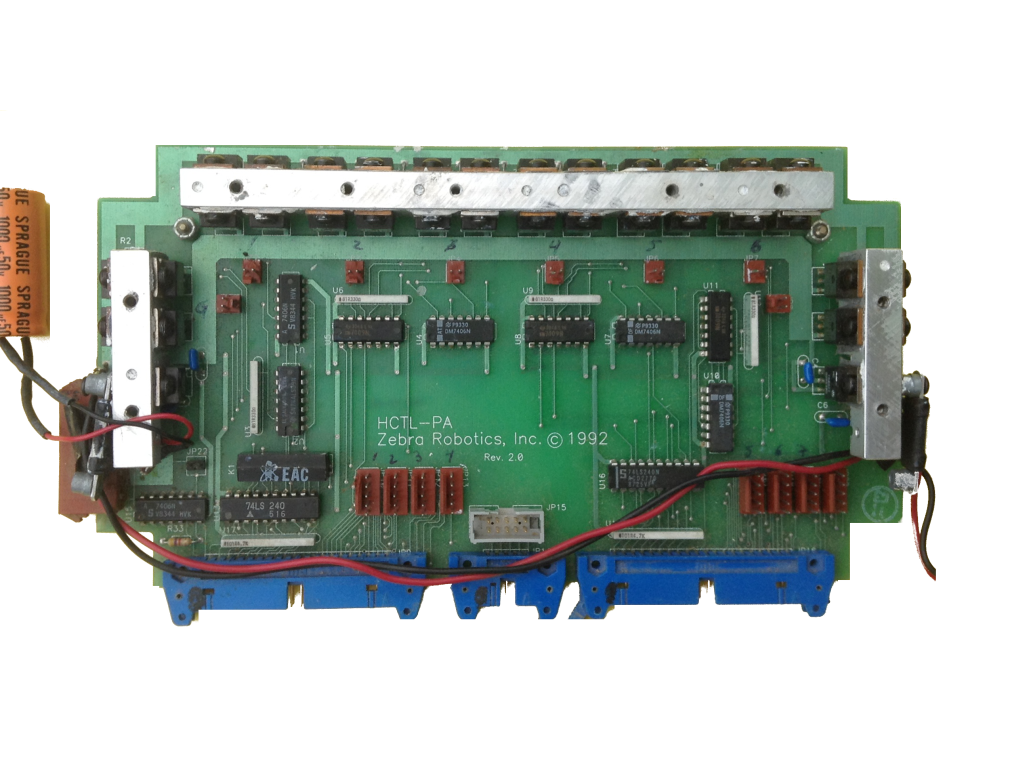
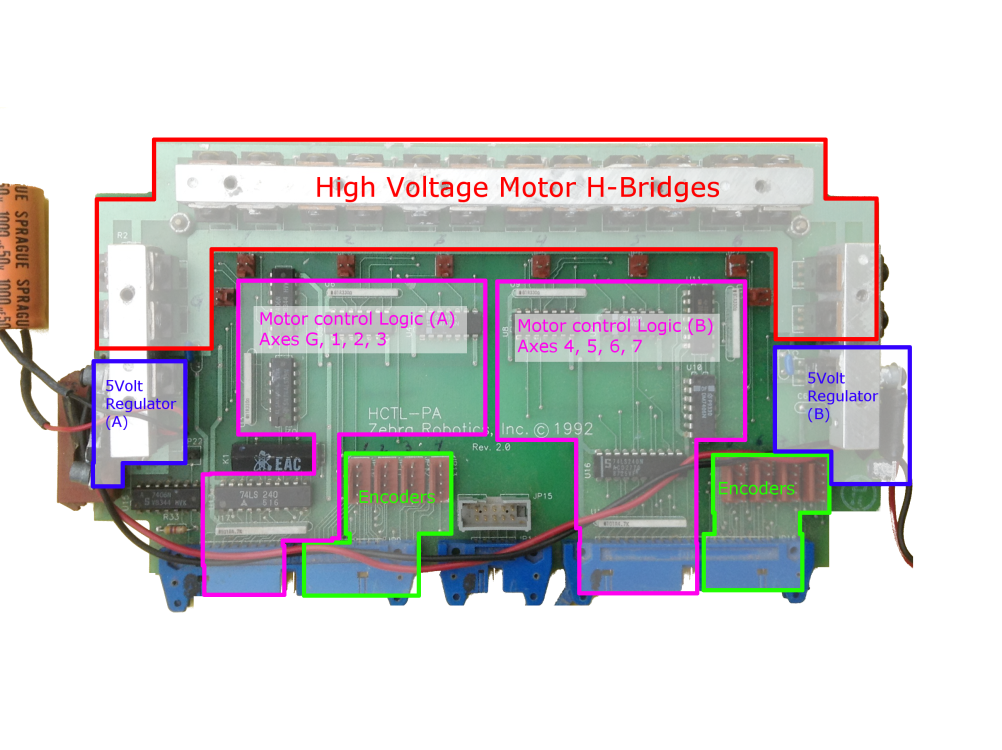
IMI Zebra Zero -- HCTL-PA -- Integrated Motion Inc. (1997)

**Overview:** Zebra is currently configured as a 7-axis robotic arm with force-feedback gripper. The arm itself has 6 built-in motors (torso, shoulder, elbow, three in the wrist), and there is a motor in the home-made add-on gripper manipulator. The base of the gripper also has a multi-axis force sensor. Each of the motors (with the possible exception of the gripper) has an optical quadrature encoder to detect motor shaft rotation. There is a control board in the base with eight H-bridge motor controllers (only 7 are currently used, one is spare), sockets for connections to the motors and encoders, and some power electronics. The base requires an external power supply, using the side 5-pin Mate-N-Lok connector. The base also has two 40-pin headers that interface with the H-bridges and encoders, and a 10-pin header that is meant to interface with add-on sensors or other circuits in the manipulator.

<https://www.tinkercad.com/things/j0VI3KPeQ3E-zebra-zero>

**Arm:** There are currently seven motors mounted within the arm, identified on the motor connectors:  
 (G) Gripper - small motor on the end of the arm  
 (1) Torso - horizontal rotation about the base  
 (2) Shoulder – vertical raising and lowering of the entire arm  
 (3) Elbow – vertical bending at the arm midpoint  
 (4) WristC – rotation of the entire differential joint near the end of the arm  
 (5) WristB – one of two motors controlling the differential joint near the end of the arm  
 (6) WristA – another of two motors controlling the differential joint near the end of the arm  
These are DC brush motors (with the possible exception of the gripper), and work on voltages up to at least 24volts. Each black shoulder tower houses one motor (motors 1 and 2). The other motors are housed in the aluminum cylinders that form the arm. There are seven barrel-type fast-blowing fuses, one for each motor, mounted inside one of the two black shoulder towers, along with wiring harnesses. Each of the motors (with the possible exception of the gripper) has an optical quadrature encoder to detect motor shaft rotation. Each encoder outputs a A/B pair of digital signals that can be used to count fractional rotation of the motor shaft. The pattern of output can be used to identify both shaft rotation direction and speed.

**Base:** The base contains a single PCB (printed circuit board). A 5-pin Mate-N-Lok connector on the side of the PCB supplies power. Original supply voltage is unknown, but is definitely higher than 5v. Probably 9 or 12 or 18 or 24volts. A 24v 7amp power supply seems to work well, but also has a tendency to explode and/or ignite the H-bridge transistors whenever anything goes wrong with the control logic. The 5volt regulators also get fairly hot when using a 24v supply. This 24v supply connects to the H-bridges and feeds the motors.

Two independent voltage regulators step down the higher voltage supply so that it can power the regular 5volt TTL motor control logic. One regulator powers the logic for axes G, 1, 2, and 3. The other regulator powers the logic for axes 4, 5, 6, and 7. The 40-pin headers provide the inputs to the control logic, which in turn enables and disables the TIP120/TIP125 transistors of the H-bridges. The inputs in the 40-pin headers have pull-up resistors, so all the inputs are 1 by default and are active low. The control logic doesn’t do much: each motor has a pair of inputs, d and v. Input d controls whether the motor runs in forward (d=z) or in reverse (d=0). Input v controls the speed of the motor using a PWM signal (v=z means no power, v=0 means full power, v=50% duty cycle means 50% power, etc.). **WARNING: DO NOT TOGGLE THE DIRECTION INPUT WITH HIGH FREQUENCY.** The v input can be toggled rapidly, at least up to a few hundred hz, to control motor speed. But the direction input, when toggled quickly, seems to cause the H-bridges to explode and/or ignite.

Encoders are not connected to the control logic at all, then are directly connected to the 40-pin headers. There is also a 10-pin header that is connected directly to a wiring harness near the gripper, but I think may not currently be used for anything.

**DEO-Nano Controller:** The DE0 and/or DE0-Nano seem ideal for interfacing with the base. Using Logisim, we can make a simple PWM circuit that outputs to the 40-pin headers on the DE0/DE0-Nano, then connect this by ribbon cable to the zebra base. Some creative wire-rearranging is necessary, since the DE0 40-pin header pinout doesn’t quite match up with the zebra’s pinout. For example, some of the DE0 pins are hard-wired to 5v, 3v, or ground, but the corresponding pins on zebra are supposed to be inputs controlling the motors or outputs from the quadrature encoders. A custom-made ribbon cable solves this. A better solution would be a pair of adapter boards with 40-pin headers and all the resistors/wiring needed to match the two pinouts. The DE0-Nano has Analog-to-Digital (ADC) circuits built in, so can be used with joysticks. Joysticks analog and digital inputs connect to the 2x13 pin header. DE0-Nano doesn’t have a built-in LCD screen, but one can be connected using some of the unused pins in the 2x13 header. Unfortunately the 2x13 pin header doesn’t output 5volts, needed for the LCD, but we can steal a 5volt signal from the 40 pin headers instead. Note: the LCD that I have is 5volt, but the DE0 outputs are 3.3volts. If we only use outputs from the DE0 to the LCD, this isn’t a problem: 3.3 volts is well above the level needed to count as a logic ON for the LCD. We can’t use outputs from the LCD, but they aren’t really needed anyway.

**Logic Levels / Voltages:** The control logic and encoders in zebra are all 5volt logic. DE0-Nano (and DE0) are 3.3v CMOS. The 5-volt pullups on zebra’s input lines could damage the DE0 fpga. For motor contr lines (output from DE0, input to zebra) a quick hack seemed sufficient: A 3.3v output from the DE0 was spliced into the pull-up circuit within the base. When the DE0 is connected, this 3.3v signal drags the pull-up reference down from 5v to a more suitable 3.3v. This is high enough to count as a logic ON for the zebra, and is low enough not to damage the DE0. The output pins from the DE0 are configured to be open-drain: they are either disabled (high impedance), or output an active low (0volt) signal. For the quadrature encoders, we should probably use a proper level-shifter, e.g. a pair of 74LVC245 chips between the zebra and the DE0. (Actually, it appears we can simply undervolt the HEDS5500 optical encoders, supplying only Vcc=3.3volts, and they still work fine but produce a 3.3v compatible output.)

**SOPC:** Altera Quartus has a SOPC (system-on-a-programmable-circuit) tool called Qsys. Using this, we can make a 100mhz, 32-bit CPU, and wire it to custom peripherals for the LCD, LEDs, Joysticks, zebra motor inputs, etc. We then write a C program (using the Altera Qsys version of Eclipse) that runs on the CPU and manages the joysticks, LCD screen, and PWM signals.

**Force sensor:** The cylindrical mount for the grip contains what appears to be a 3-axis (or 4-axis?) force sensor. The circuitry in the grip base is wired directly to the 10-pin header at the center of the robot base. Pin 8 appears to have a broken connection somewhere between the base and the elbow. The interface protocol for accessing the force sensor is unknown, but it probably involves a simple 5-volt TTL multiplexor and accessing a simple ADC to read the strain gauges.

**DE0 13x2 expansion bus:** This is used for joystick ADC and buttons (inputs), and for controlling the LCD. There aren’t enough general purpose I/O pins for all eight joysticks (since we use some GPIO pins for the LCD output), so we need to use two of the dedicated clock input pins for buttons. This is fine, except we can’t use the internal weak pull-up resistors on those pins, so we need external pullups on those two pins.

**TO DO:** Some things to be done in future, from lowest level to highest level…

* Build a different / better manipulator end for the arm, and/or add sensors to the arm.
* Build a better case to hold the electronics.
* Replace, upgrade, and/or repair the control board in the base. The H-Bridges are terribly inefficient and explode more often than I would prefer. There is no dynamic or regenerative braking, and apparently no proper handling of voltage spikes caused by PWM. The H-bridges are missing protection diodes. The 5volt regulators are terribly inefficient and get very hot – a switching regulator would be much better. The control logic is barely sufficient to run the H-bridges, a more sophisticated circuit would be nice, e.g. including current-sensing and over-current protection, proper dynamic braking, etc.
* Build adapter boards instead of crazy ribbon cables.
* Figure out how to access the force sensor strain gauges, and fix broken wire on pin 8.
* Hook up the quadrature encoders to the DE0.
* Make a quadrature-encoder peripheral in the DE0, and connect it to the CPU.
* Modify C program to track encoders and provide better control to motors, e.g. hold steady even in presence of gravity or other forces, detect hard-stops when rotating too far, etc.
* Add simple acceleration/deceleration control.
* Add simple motion planning.
* Add goal-based motion planning.
* Add vision or sensor feedback.