Appendix A

6.270 Hardware

This chapter is partly tutorial and partly technical reference: in additional to documenting the 6.270 hardware, it explains the design in a way that would be understandable to the beginner. The discussion does however assume familiarity with some ideas of digital electronics.

The information presented here should be considered optional, as it is not strictly necessary to know it to build a robot. Hopefully though, this chapter will satisfy most readers' curiosity about how the 6.270 hardware works.

A.1 The Microprocessor and Memory

At the most primitive level, a computer consists of a microprocessor, which executes instructions, and a memory, in which those instructions (and other data) is stored.

Figure A.1 shows a block diagram of these two components. The diagram shows four types of wires that connect the microprocessor and the memory:

Address Bus. These wires are controlled by the microprocessor to select a particular location in memory for reading or writing.

The 6.270 board uses a memory chip that has 15 address wires. Since each wire has two states (it can be a digital one or a zero), 2 to the 15th power locations are possible. 2^{15} is precisely 32,768 locations; thus, the system has 32K of memory.

Data Bus. These wires are used to pass data between the microprocessor and the memory. When data is written to the memory, the microprocessor drives these wires; when data is read from the memory, the memory drives the wires.

In our example (and in the 6.270 board), there are eight data wires (or bits). These wires can transfer 2^8 or 256 different values per transaction. This data word of 8 bits is commonly referred to as a *byte*.

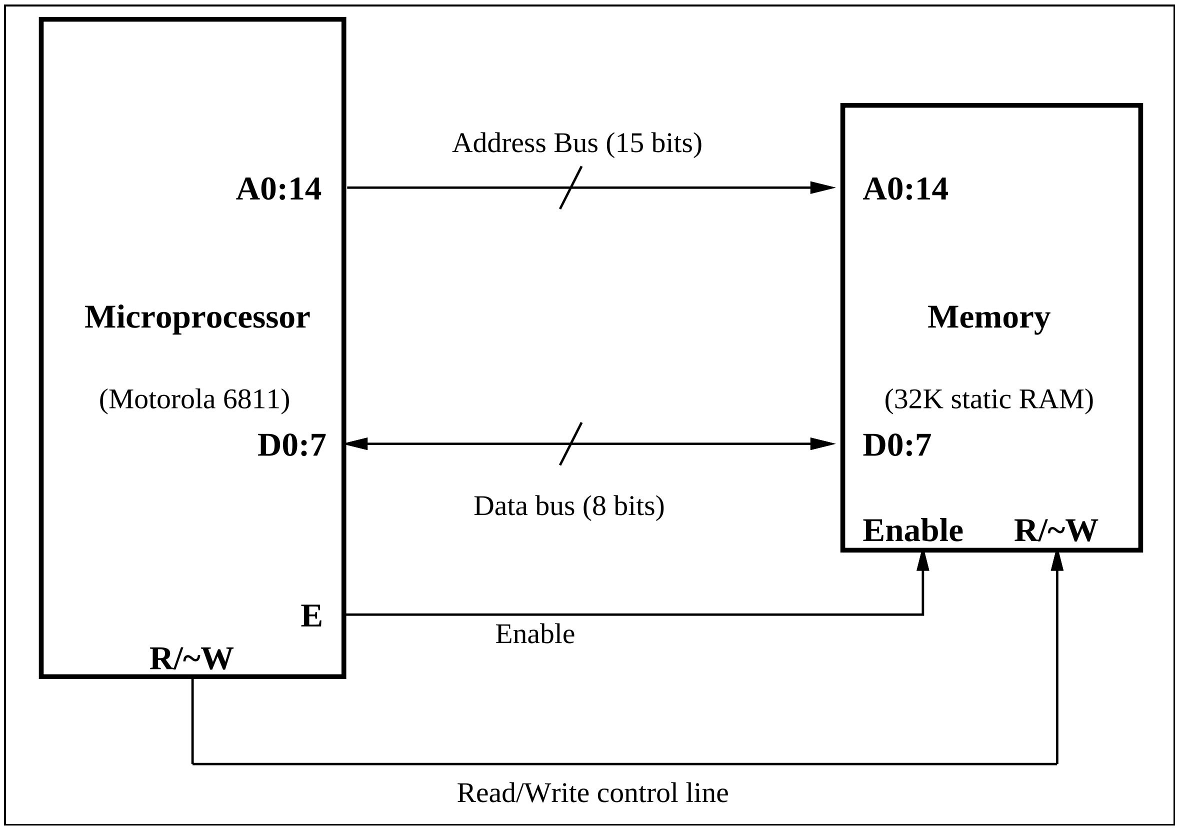

Figure A.1: Block Diagram of Microprocessor and Memory

Read/Write Control Line. This single wire is driven by the microprocessor to control the function of the memory. If the wire is logic true, then the memory performs a "read" operation. If the wire is logic zero, then the memory performs a "write operation."

Memory Enable Control Line. This wire, also called the *E clock*, connects to the enable circuitry of the memory. When the memory is enabled, it performs either a read or write operation as determined by the read/write line.

A.1.1 Multiplexing Data and Address Signals

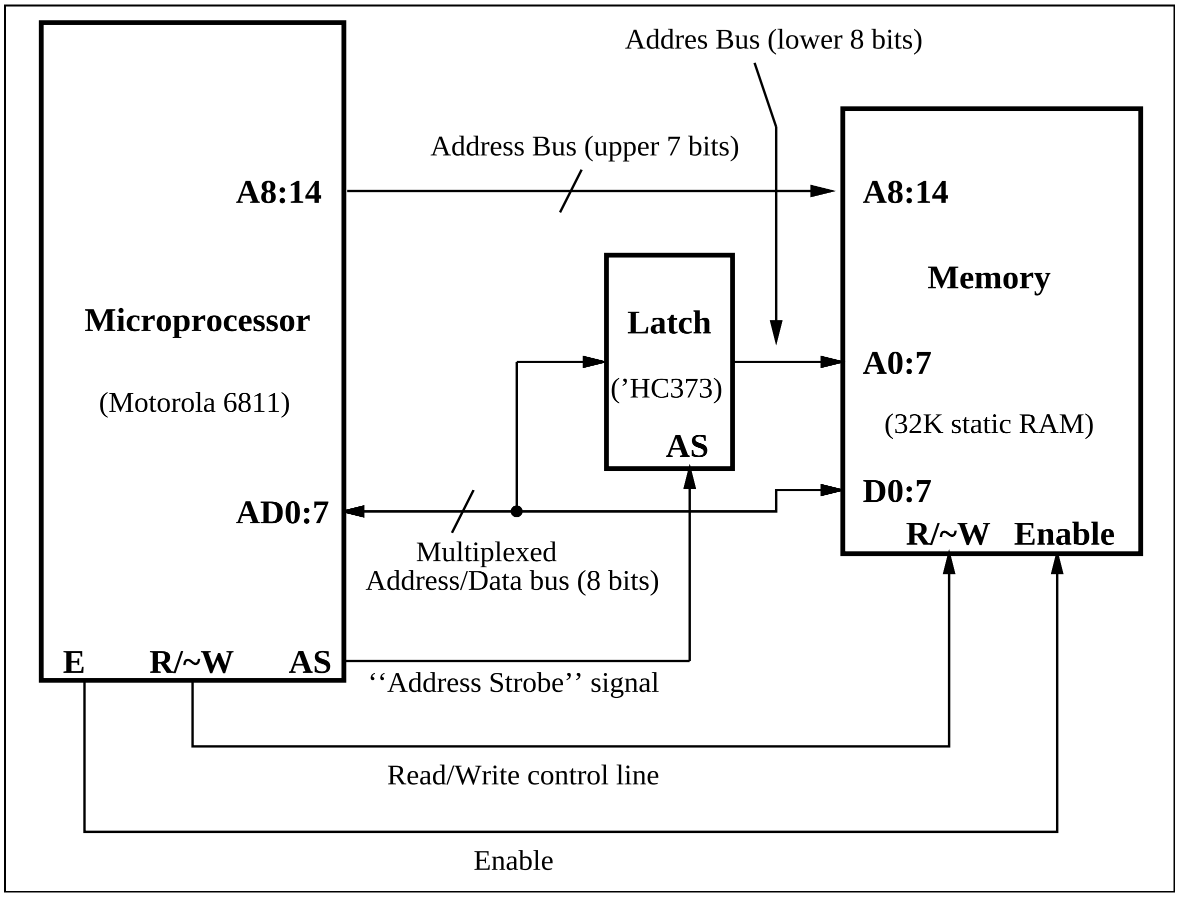

Figure A.2: Block Diagram of Microprocessor and Memory with Latch

Things are a little more complex with the particular microprocessor that is used in the 6.270 board, the Motorola 6811. On the 6811, The eight data bus wires take turns functioning as address wires as well.

When a memory location is needed (for reading or writing), first the data wires function as address wires, transmitting the eight lower-order bits of the address. Then they function as data wires, either transmitting a data byte (for a write cycle) or receiving a data byte (for a read cycle). All this happens very fast; 2 million times per second to be exact.

The memory needs to help to deal with the split-personality data/address bus. This help comes in the form of an *8-bit latch.* This chip (the 74HC373) performs the function of latching, or storing, the 8 address values so that the memory will have the full 15-bit address available for reading or writing data.

Figure A.2 shows how the latch is wired. The upper 7 address bits are normal, and run directly from the microprocessor to the memory. The lower 8 bits are the split-personality, or, more technically, *multiplexed* address and data bus. These wires connect to the inputs of the latch and also to the data inputs of the memory.

An additional signal, the *Address Strobe* output of the microprocessor, tells the latch when to grab hold of the address values from the address/data bus.

When the full 15-bit address is available to the memory (7 bits direct from the microprocessor and 8 bits from the latch), the read or write transaction can occur. Because the address/data bus is also wired directly to the memory, data can flow in either direction between the memory and the microprocessor.

This whole process—the transmitting of the lower address bits, the latching of these bits, and then a read or write transaction with the memory—is orchestrated by the microprocessor. The E clock, the Read/Write line, and the Address Strobe line perform in tight synchronization to make sure these operations happen in the correct sequence and within the timing capacities of the actual chip hardware.

A.2 Memory Mapping

So far we have seen how a memory can be connected to the address space of a microprocessor. In a circuit like the one of the 6.270 board, the microprocessor must interact with other devices than the memory—for example, motors and sensors.

A typical solution uses 8-bit latches for input and output. These latches are connected to the data bus of the microprocessor so that they appear like a location in memory. Then, the act of reading or writing from one of these memory locations causes data to be read from or written to a latch—to which the external devices are connected.

Figure A.3 is a block diagram of the 6.270 Robot Controller Board system. Following the present discussion that concerns how the motors and sensors are addressed by the microprocessor, notice that a chip labelled "273" is connected to the data bus. The '273 has outputs that control the motors (through chips labelled "L293," which will be discussed later). The digital sensors are driven into the data bus by a chip

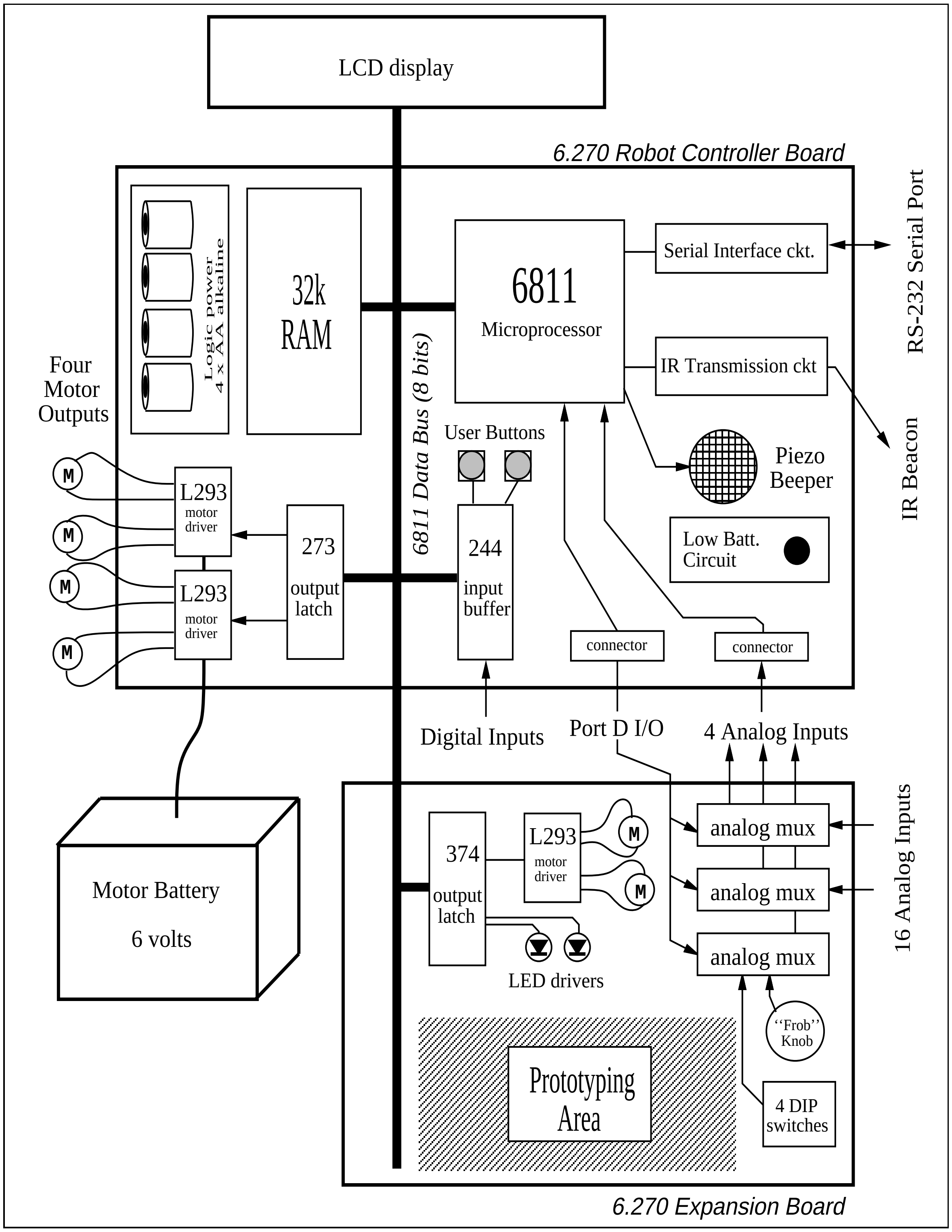

Figure A.3: 6.270 System Block Diagram

labelled "244." On the expansion board, a 374 chip, another output latch, is used for eight bits of digital output.

These interface latch chips are used in a technique called *memory mapping*. The chips are "mapped" to a particular address in the microprocessor's memory.

The following discussion will show how both the 32k RAM memory and the digital input and output latch chips share the address space of the microprocessor.

A.2.1 Memory-Mapping the RAM

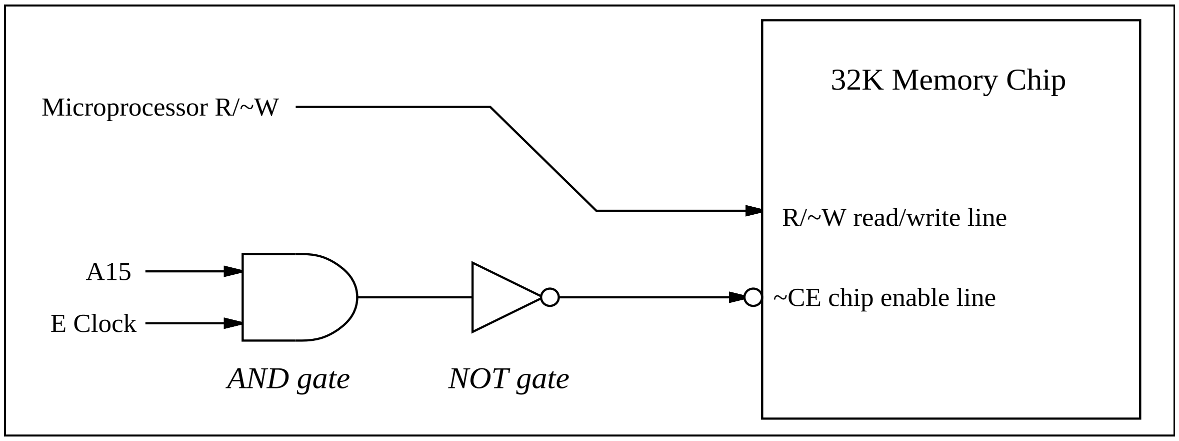

Figure A.4: Enabling the Memory

The 6811 has a total of 16 address bits, yielding 64K bytes of addressable locations (65536, to be exact). Half of this space will be taken up by the 32K memory chip (also known as a *RAM* chip, for "random access memory").

The 6811 has a bank of *interrupt vectors*, which are hardware-defined locations in the address space that the microprocessor expects to find pointers to driver routines. When the microprocessor is reset, it finds the reset vector to determine where it should begin running a program.

These vectors are located in the upper 32K of the address space. Thus, it is logical to map the RAM into this upper block, so that the RAM may be used to store these vectors.

The technique used to map the memory to the upper 32K block is fairly simple. Whenever the 6811's A15 (the highest-order address bit) is logic one, an address in the upper 32K is being selected. The other fifteen address bits (A0 through A14) determine that address.

A logic gate is used to enable the memory when A15 is logic one *and* when the E clock is high (since the E clock must control the timing of the enable). Figure A.4 shows a block diagram of this circuit. (The actual circuit to enable the RAM, shown in Figure A.8, is slightly more complex due to considerations of battery-protecting the memory, as explained later.)

Memory chips are part of a class of chips that have *negative true* enable inputs. This means that they are enabled when the enable input is logic zero, not logic one.

There are two methods for denoting an input that is negative true. As shown in Figure A.4, the chip enable input is shown with connecting to a circle. This circle indicates a negative true input. Also, the name for the signal, *CE* is prefixed with a ~ symbol.

The function of the NOT gate shown in the diagram is to convert the positive-true enable produced by the AND gate into the negative-true signal required by the ~CE input. (Often these two gates are collapsed into a single NAND gate.)

A.2.2 Memory-Mapping with the 74HC138 Chip

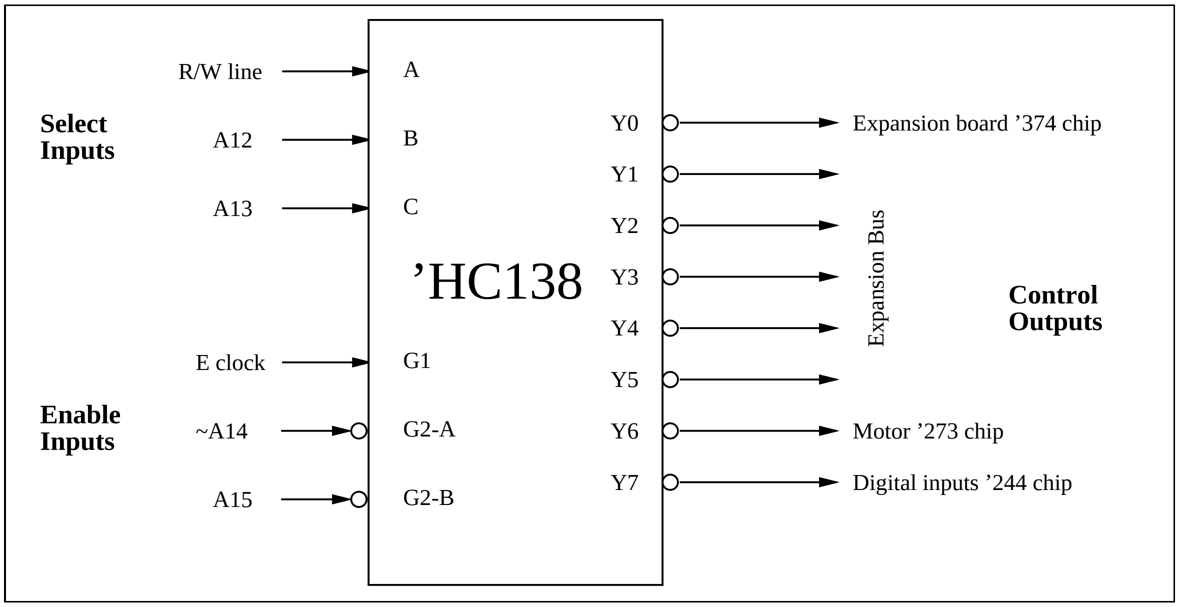

Figure A.5: Wiring the 'HC138 Address Decoder

Figure A.5 shows the 74HC138 chip, which is commonly used in circuits that map devices onto an address space. This chip is a 3-to-8 decoder: a binary number of three digits (the *select inputs*) causes one of eight possible outputs to be selected (the *control outputs*). The chip also has three *enable inputs*, all of which must be enabled to make the chip become active.

The outputs of the '138 chip control the input and output latches shown in the system block diagram. The '138 determines when these latches are activated, either to read data from the data bus (in the case of the '273 output latch), or to write data onto the data bus (in the case of the '244 input latch).

Enable Inputs

The enable inputs of the '138 determine *when the chip will become active,* and thereby turn on one of the input or output latches. These enables inputs are critical because the '138 must not become active at the same time as the RAM chip. In it did, then two devices (the RAM and perhaps a '244) would attempt to drive the data bus simultaneously, causing a problematic situation called *bus contention.*

As shown in Figure A.5, A15, the highest order address bit, is connected to a *negative enable* of the '138. Thus A15 must be *zero* to enable the chip. Since the RAM is enabled only when A15 is one (as was explained earlier), there is no chance that the '138 and the RAM could be active at the same time.

~A14, which is the logical inverse of A14, is connected to a second negative enable of the '138. Thus when A14 is one, ~A14 is zero, and the G2-A enable is true. So A14 must be one in order to active the '138.

The final enable input is positive true, and is connected to the 6811 E clock. When A15 is zero and A14 is one, the E clock will turn on the '138 at the appropriate time for standard 6811 read/write cycles.

Select Inputs

Given that the '138 is enabled, the A, B, and C inputs determine which device connected to its outputs will be activated. A, B, and C form a binary number (C is the most significant bit) to determine the selected output.

The A13 and A12 address bits and the 6811 read/write line make the selection. Suppose A13 and A12 are one. The read/write line makes the final choice. This line is one for a read and zero for a write. If a read operation is in progress, then the ABC inputs will form the number 7, and the Y7 output will be activated. As shown in Figure A.5, this output connects to the digital input '244 chip. So, the '244 chip will turn on and will drive a byte onto the data bus. The read operation will complete with this byte having been read from the location in 6811 address space that was selected.

Notice that address bits A0 through A11 have no effect on the operation just described. As long as A15 is zero, A14, A13, and A12 are one, a read operation will cause the '138 to turn on the digital input '244 chip to write a byte onto the data bus. Thus, the digital input chip is selected by a read from any address from \$7000 to \$7FFF[1]. This is fairly wasteful of the address space of the 6811, but keep in mind

[1]These numbers are expressed in the hexadecimal numbering system, in which each digit represents a four-bit value from zero (0) to fifteen (F)

that the only circuitry required to arrange this solution was the '138 chip.

Suppose a write operation were to occur in that same range of memory. The relevant upper four address bits would have the same values, but the read/write line would be zero (indicating the write operation). Thus the '138 ABC inputs would form the number 6, and output Y6 would be activated. Y6 is connected to the '273 chip that controls the motors; thus, the '273 would latch the value present on the data bus during the write operation.

As shown in Figure A.5, most of the '138 outputs are still available for future expansion. The 6.270 Expansion Board includes a circuit with one '374 chip, connected to the Y0 output. Outputs Y1 through Y5 are left free for further expansion use.

A.2.3 System Memory Map

Figure A.6 summarizes the memory map solution that has been implemented for the 6.270 Board.

The 32K RAM takes up half of the total address space of the microprocessor. As indicated in the map, it is located in the upper 32K of the microprocessor's memory, from addresses $8000 to $FFFF.

The four digital input and output ports are mapped at locations starting at $4000, $5000, $6000, and $7000.

There is small area of memory that is internal to the 6811 chip itself. This memory consists of 256 bytes located at the start of the address space, from locations $00 to $FF.

The 6811 also has a bank of 64 internal *special function registers*, located at addresses $1000 to $103F. These registers control various hardware features of the 6811 (the analog inputs and serial communications are two examples).

The remainder of this section presents details on the digital input and output circuit wiring.

A.2.4 Digital Inputs

Figure A.7 shows the digital input circuitry. U6, a 74HC244 chip, is used to latch an eight-bit word of sensor inputs and drive the 6811 data bus with that value when the chip is selected.

The '244 chip has two halves which may be separately enabled. The Y7 select is connected to both enable inputs, so that both halves of the chip are always selected simultaneously.

The lower two bits of the '244 are connected to the two user buttons (which have been dubbed CHOOSE and ESCAPE). The upper six bits are connected to the digital input header.

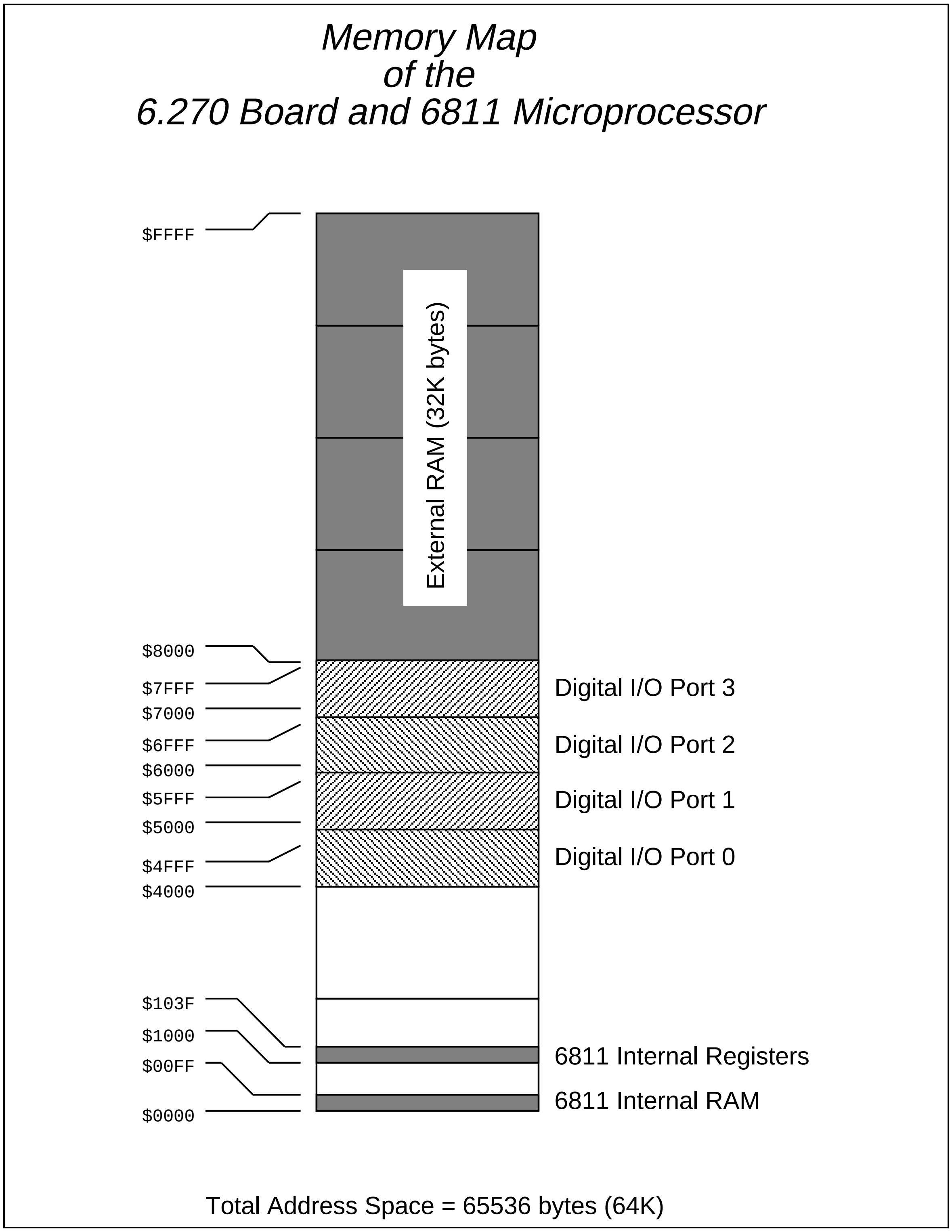

Figure A.6: 6811 System Memory Map

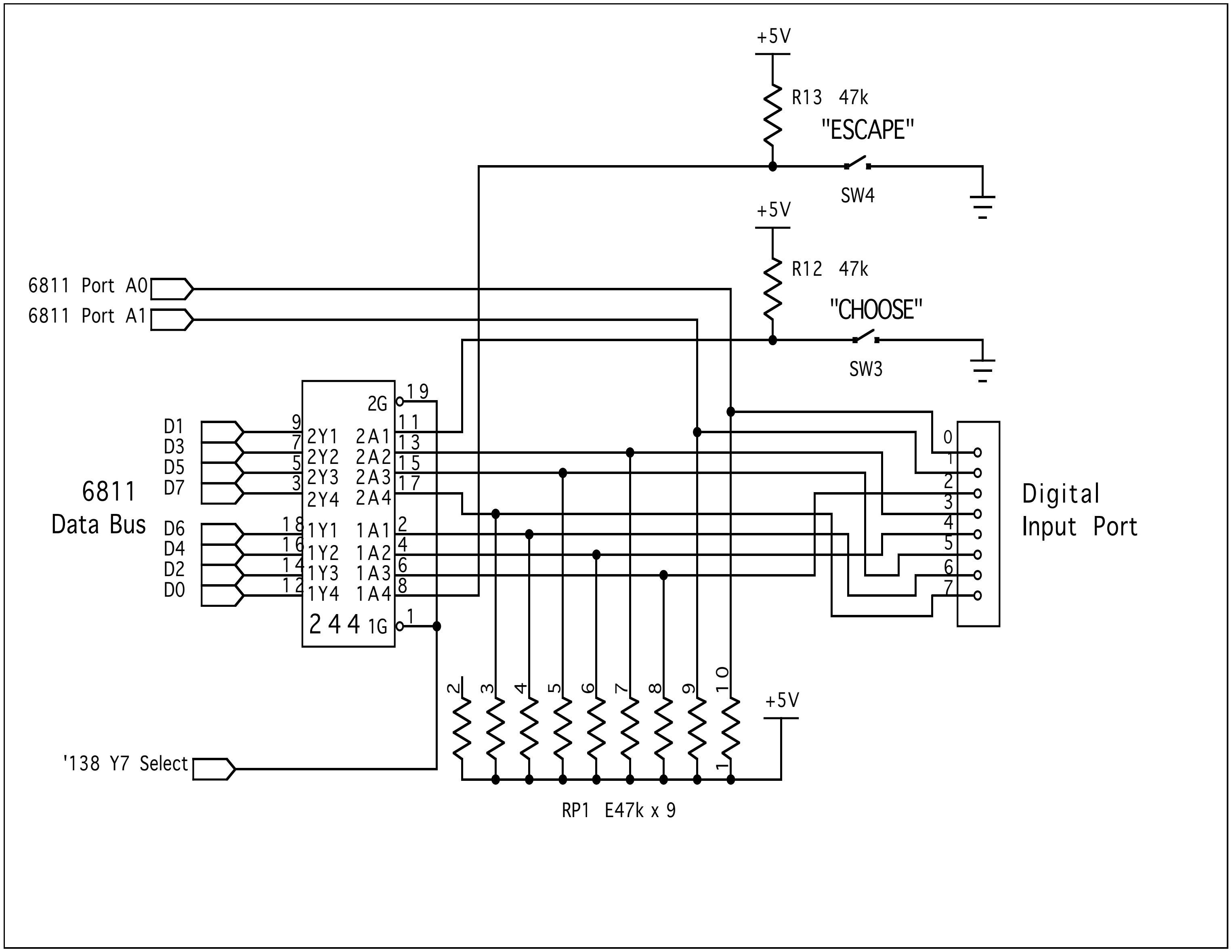

Figure A.7: Digital Input Circuit

The lower two bits of the input header are connected to two timer inputs inputs of the 6811. These inputs can be used to precisely measure waveforms, or can simply be used for digital input. If shaft encoding is used, it is the input capture functions on these two input ports which are used for the encoding. The library functions written to perform digital inputs insulate the user from the fact that the eight pins on the input header are not mapped contiguously to one location in memory.

RP1, a 47K resistor pack, acts as pull-up resistors to the inputs of the '244 chip, making the default values of the inputs one.

A.2.5 Digital Outputs

Figure A.13 shows the complete schematic for the '273 output latch controlling the motors. For the purpose of the discussion to this point, notice that the data inputs '273 are connected to the 6811 data bus. The Y6 select signal connects to the clock input of the '273; when Y6 is activated, the '273 latches the value present on the data bus.

The outputs of the '273 connect to the motor driver chips. This circuitry is explained in the following section.

Figure A.15 is the schematic of the motor circuit present on the 6.270 Expansion Board.

A.2.6 6811 and Memory Schematic

Figure A.8 presents the schematic of the 6811, memory, address decoding, and supporting main circuitry on the 6.270 Processor Board. By the end of this chapter, most of the circuitry depicted here will be explained.

A.3 The Motor Drivers

Motors are high-powered devices in the world of digital electronics. A typical digital output can supply about 10 to 20 milliamperes (mA) of current; a small permanent-magnet motor requires anywhere from 500 to 4000 mA of current. It should not come as a surprise that special circuitry is required to drive motors.

A.3.1 The H-Bridge Circuit

A circuit known as the *H-bridge* (named for its topological similarity to the letter "H") is commonly used to drive motors. In this circuit (depicted in Figure A.9), two of four transistors are selectively enabled to control current flow through a motor.

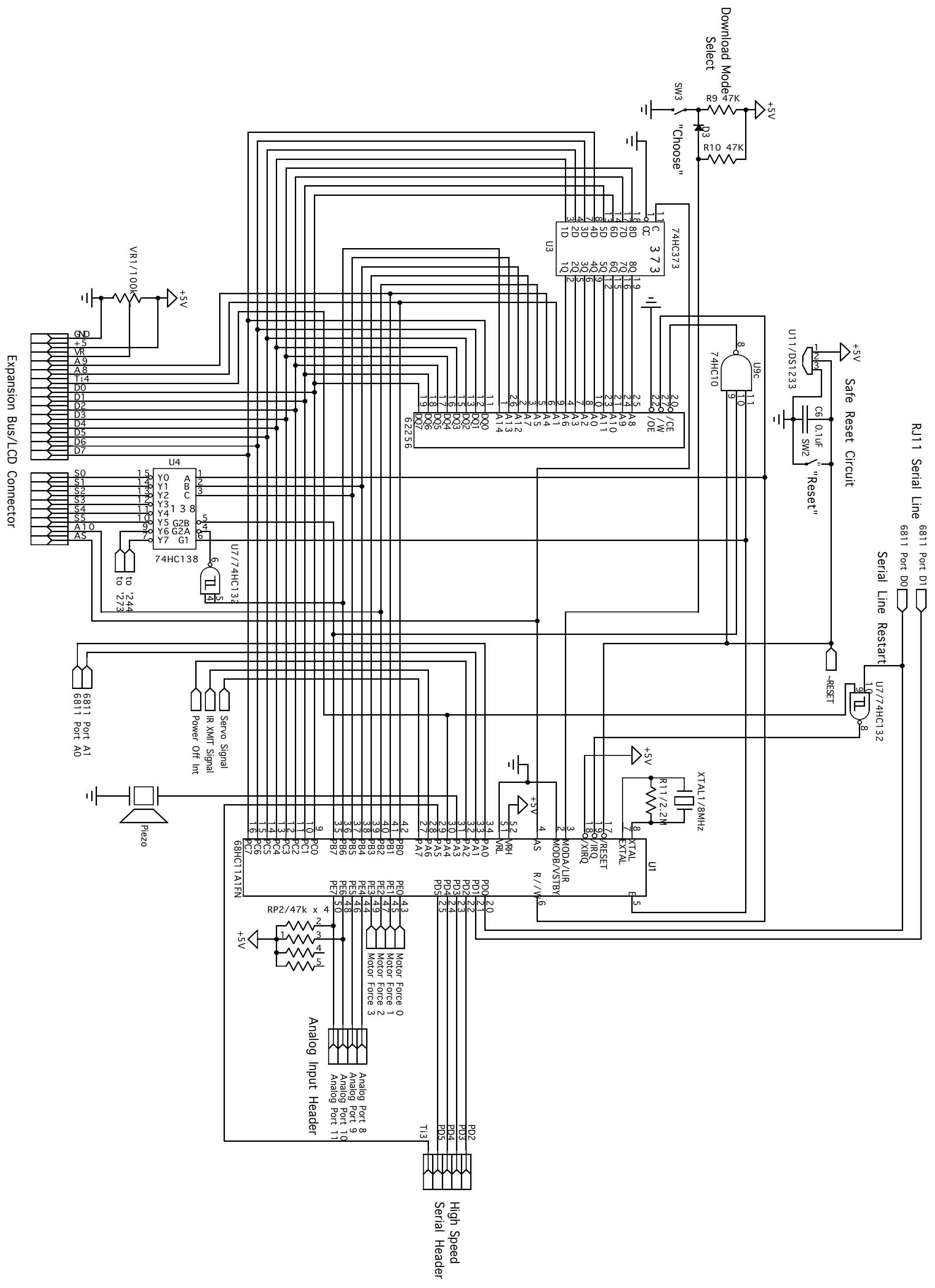

Figure A.8: 6811, Memory, Address Decoding and Miscellaneous Circuitry

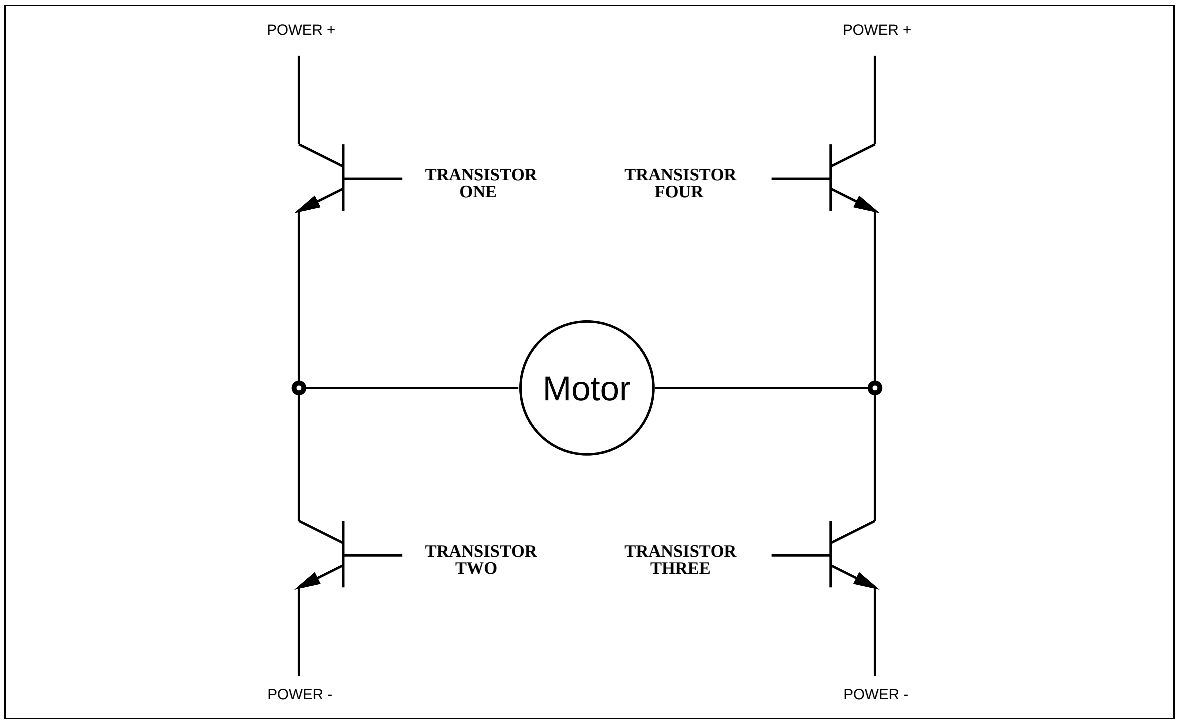

Figure A.9: The H-Bridge Circuit

As shown in Figure A.10, an opposite pair of transistors (Transistor One and Transistor Three) is enabled, allowing current to flow through the motor. The other pair is disabled, and can be thought of as out of the circuit.

By determining which pair of transistors is enabled, current can be made to flow in either of the two directions through the motor. Because permanent-magnet motors reverse their direction of turn when the current flow is reversed, this circuit allows bidirectional control of the motor.

A.3.2 The H-Bridge with Enable Circuitry

It should be clear that one would never want to enable Transistors One and Two or Transistors Three and Four simultaneously. This would cause current to flow from Power+ to Power− through the transistors, and not the motors, at the maximum current-handling capacity of either the power supply or the transistors.

To facilitate control of the H-bridge circuit, enable circuitry as depicted in Figure A.11 is typically used.

In this circuit, the inverters ensure that the vertical pairs of transistors are never enabled simultaneously. The *Enable* input determines whether or not the whole circuit is operational. If this input is false, then none of the transistors are enabled,

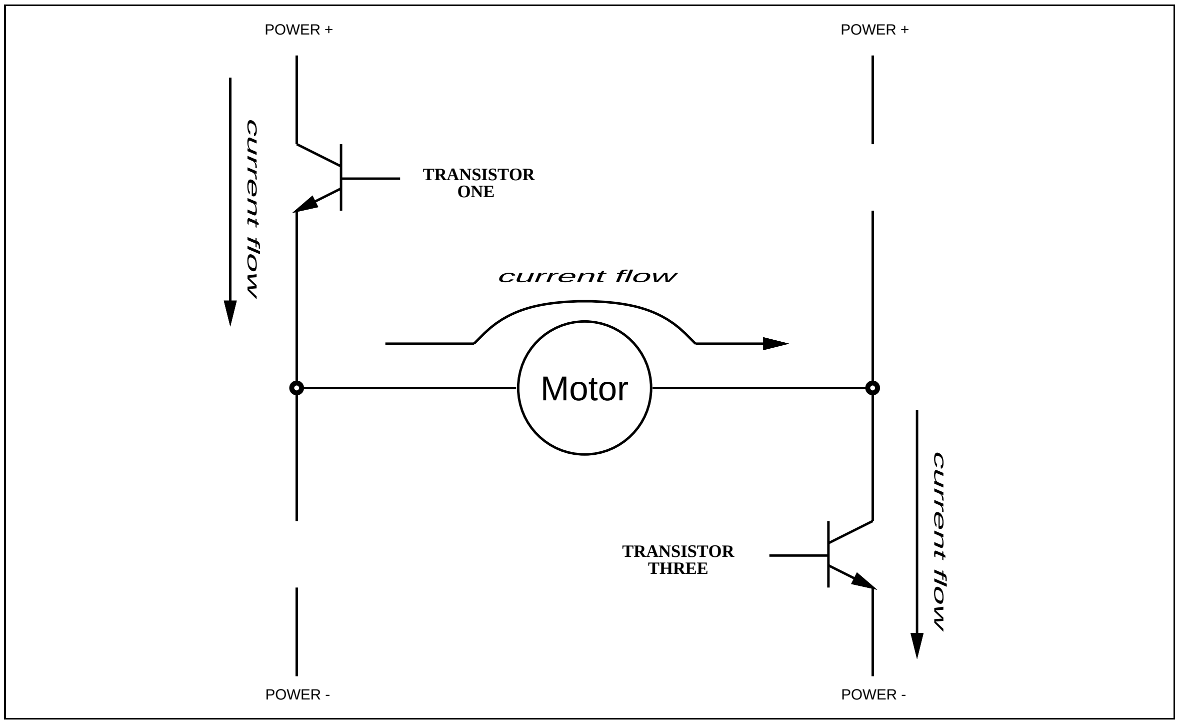

Figure A.10: The H-Bridge with Left-to-Right Current Flow

and the motor is free to coast to a stop.

By turning on the *Enable* input and controlling the two *Direction* inputs, the motor can be made to turn in either direction.

Note that if both direction inputs are the same state (either true or false) and the circuit is enabled, both terminals will be brought to the same voltage (Power+ or Power−, respectively). This operation will actively brake the motor, due to a property of motors known as *back emf*, in which a motor that is turning generates a voltage counter to its rotation. When both terminals of the motor are brought to the same electrical potential, the back emf causes resistance to the motor's rotation.

A.3.3 The SGS-Thomson Motor Driver Chip

A company named SGS-Thomson makes a series of chip called the L293 that incorporates two H-bridge motor-driving circuits into a single 16-pin DIP package. Figure A.12 shows a block diagram of this incredibly useful integrated circuit.

The schematic of the motor circuit (Figure A.13) shows how the L293 chips are used in the 6.270 board design. Eight bits are used to control four motors. Four of the bits determine the direction of the motors (with the assistance of inverters) and four bits determine the when the motors are on or off.

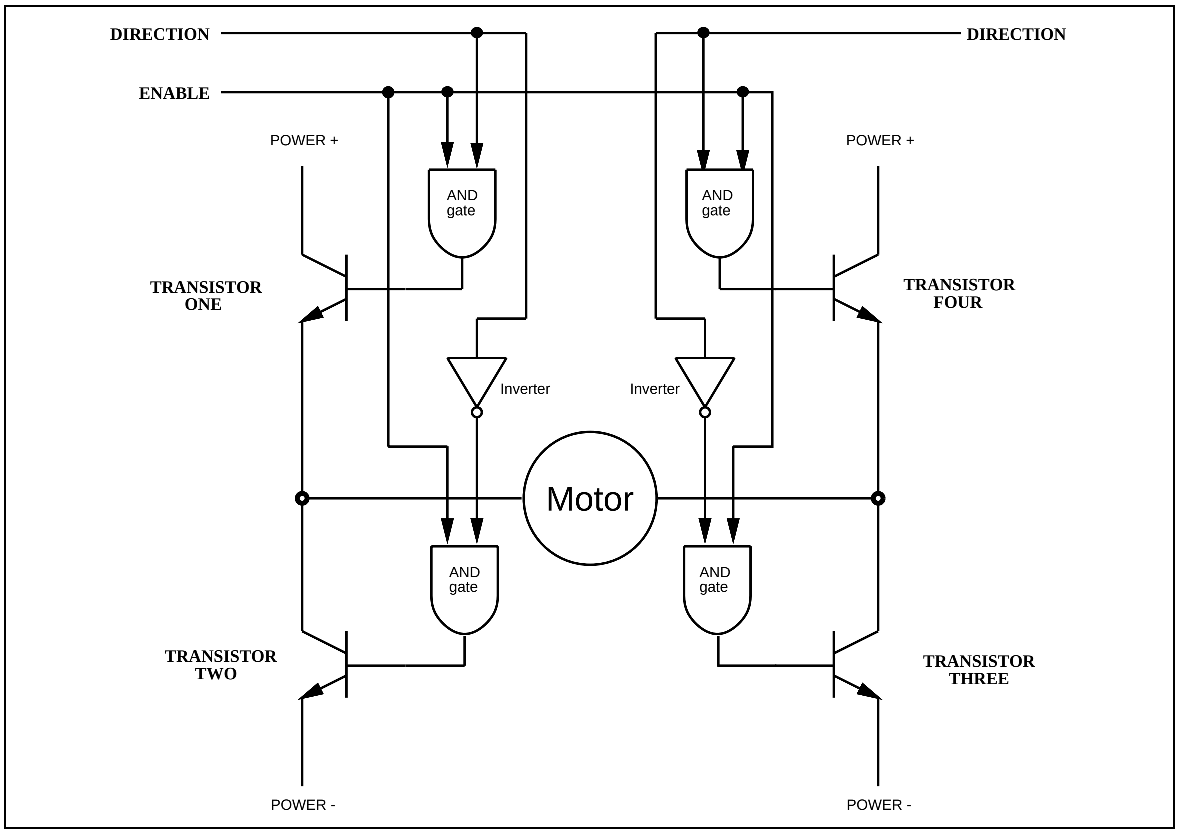

Figure A.11: The H-Bridge with Enable Circuitry

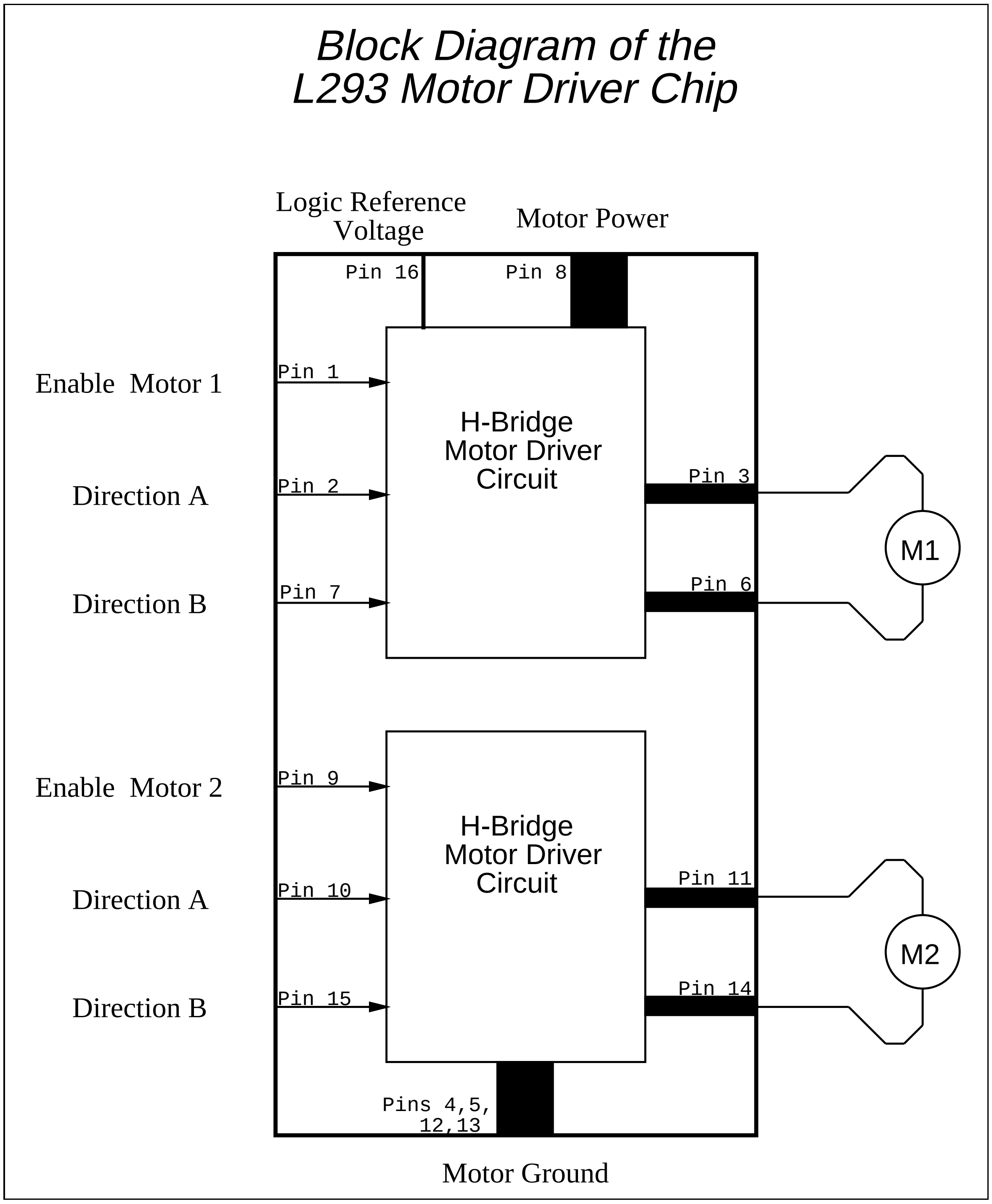

Figure A.12: The SGS-Thomson L293 Motor Driver IC

Notice that braking a motor is not possible with this circuit configuration, because the inverters do not allow both direction inputs of a given motor to be the same state.

The speed of a motor may be controlled by pulsing its enable bit on and off. This technique, called *pulse width modulation*, is explained in the chapter on motors.

A.3.4 Power Considerations

Current Handling and Spike Protection

In the 6.270 circuit design, two L293 chips are used in parallel to control each motor. This is an unconventional circuit hack add to the current-handling capacity of the motor drivers.

Two different L293 chips are used in this circuit. One chip, the L293D, has internal spike-protecting diodes on the motor outputs. These diodes protect the motor chip and the rest of the circuit from electrical noise generated by the motors. The other chip, the L293B, does not have these diodes, but has a greater current handling ability than the 'D chip.

The L293D can supply 600 mA of current per channel; the L293B, 1000 mA. Used in parallel, the circuit can supply 1600 mA per channel. Because of the spike-killing diodes contained in the 'D chip, the overall circuit is safe to use.

Power Supply Isolation

The electrical noise generated by motor can be hazardous to a microprocessor circuit even with the use of the diodes. For this reason, separate power supplies are used for the motors and the rest of the microprocessor electronics.

Figure A.14 shows the power-supply circuitry. Notice that *Logic Power*, for the microprocessor circuitry, is a configuration of four AA cells, while + *Motor*, power for the motors, is supplied through the J1 connector.

The motor ground and the logic ground must be kept at the same potential so that the control signals from the '273 chip shown in Figure A.13 can communicate with the L293 chips. These grounds are kept at the same potential by the inductor L1.

The inductor is used to provide reactance (frequency-dependent resistance) to trap spikes that might travel from the motors, through the L293 chips, and into the microprocessor circuit.

A.3.5 Expansion Board Motor and LED Circuitry

The 6.270 Expansion Board plugs into the Expansion Bus header depicted in Figure A.8. This header connects to the 6811 data bus and to the six '138 select signals that are not used on the main board.

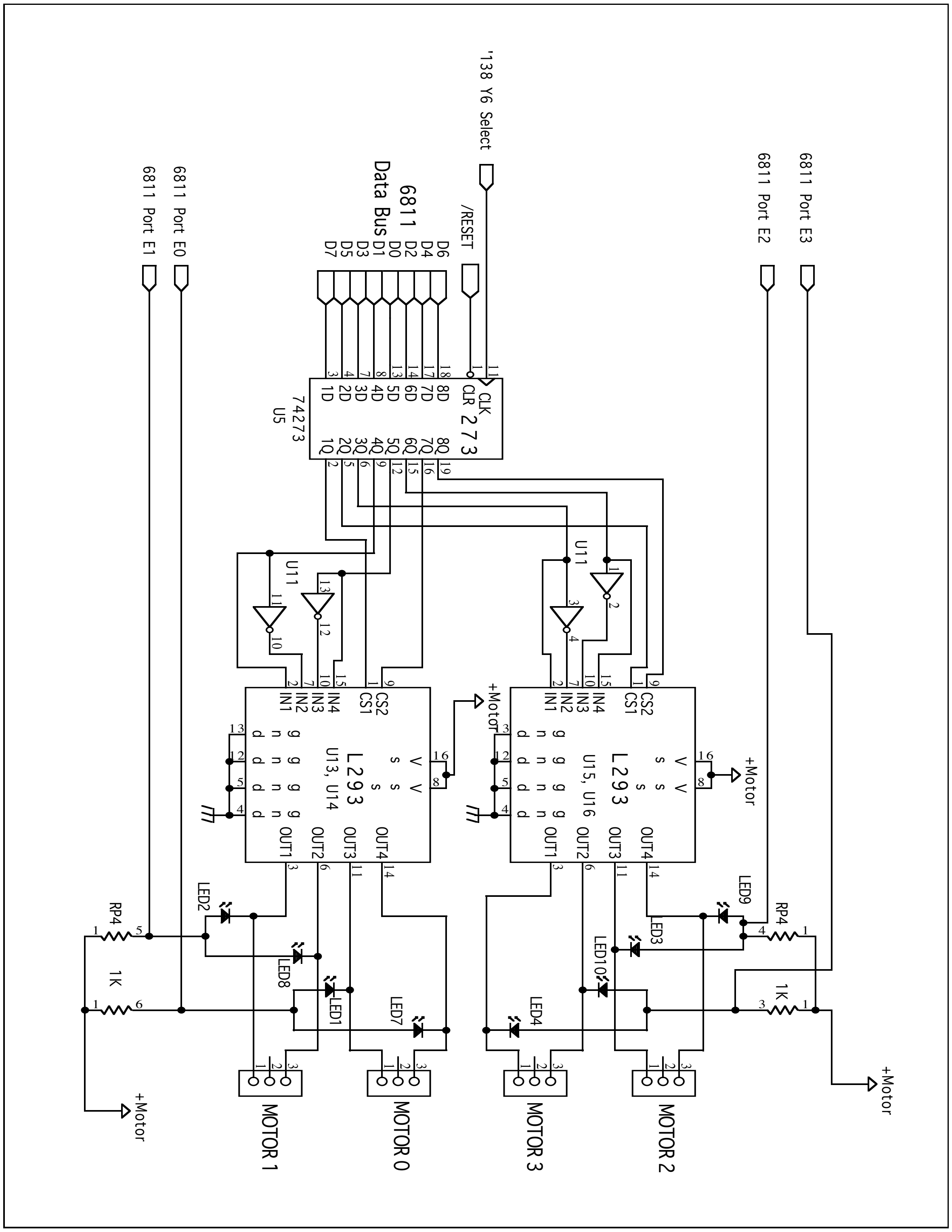

Figure A.13: Motor Driver Circuit

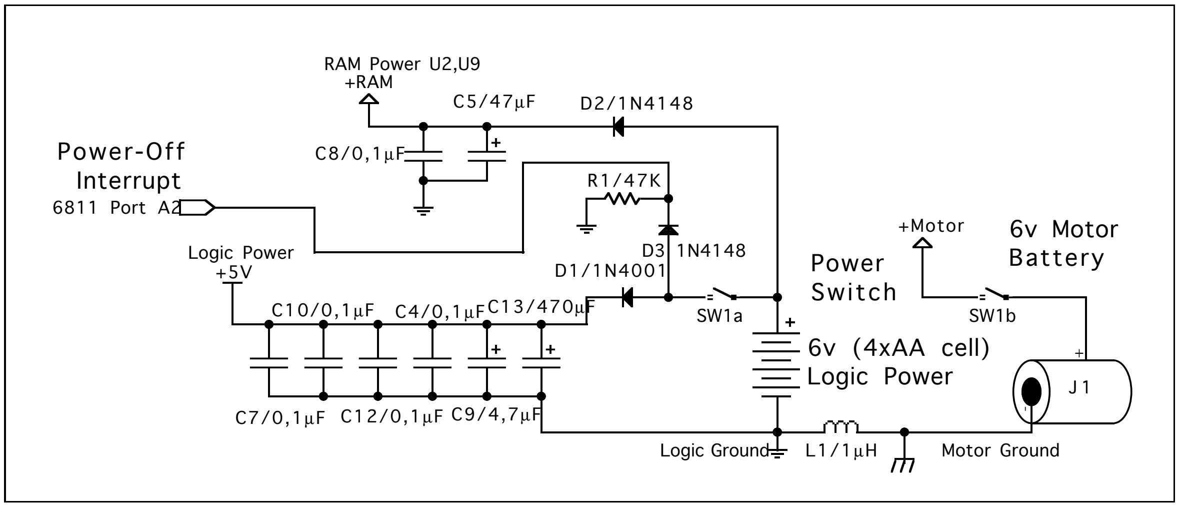

Figure A.14: Power Filtering and Switching Circuit

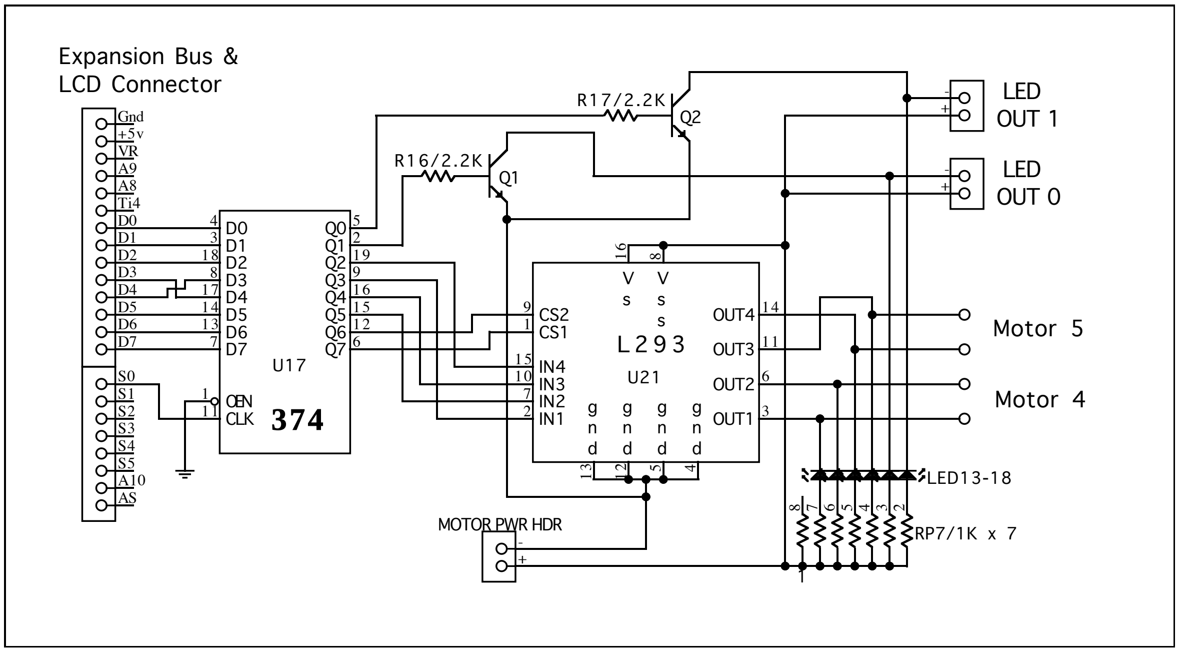

Figure A.15: Expansion Board Motor and LED Circuitry

Figure A.15 illustrates how a single L293D chip is used on the Expansion Board to provide outputs for two additional motors. Because six outputs of the '374 chip are wired to control all four direction inputs and the two enable inputs of the L293D, the motors can be braked if desired. Or, four unidirectional devices may be powered.

The remaining two bits of the '374 are connected to transistor drivers. These transistor circuits are well-suited for powering light-load devices, such as LEDs.

A.4 Analog Inputs

The 6811 has on-chip circuitry to perform an analog-to-digital signal conversion. In this operation, a voltage from 0 to 5 volts is linearly converted into an 8-bit number (a range of 0 to 255). This feature is one of the many that make the 6811 very well suited for control applications.

The 6811 has eight of these analog inputs. In the 6.270 board design, four of these pins are wired to a *motor current monitoring circuit*, and four of them are wired to input connectors.

A.4.1 Motor Current Monitoring Circuit

When the L293 chips drive a motor, there is a voltage drop across the transistors that form the H-bridge. The transistor connected to motor ground (0 volt potential) might drive the motor at some voltage between .2 and .8 volts; the transistor connected to the positive terminal of the battery (say it's at 6 volts) might drive the motor between 5.2 and 5.8 volts.

The amount of this voltage drop is proportional to the amount of current being supplied by the motor-driving transistor. When more current is being supplied, the transistor drops more voltage.

This undesirable property of the L293 transistors is exploited to give a crude measurement of the amount of current being driven through the motor. A fundamental property of motors is that as the amount of work they are performing increases, the amount of current they drawn also increases. So the current measurement yields data on how hard the motor is working—if it is turning freely, if it is stalled, or if it is working somewhere in between.

As indicated in Figure A.13, the voltage feedback point is tapped from the indicator LEDs that are connected to the motor outputs. The voltage across the LEDs will decrease as a result of increased current draw of the motor (and the corresponding decreased performance of the L293's). This voltage is fed to a 6811 analog input and can be measured by the 6811 analog-to-digital conversion hardware.

Each of the four motor circuits is wired in this way to a 6811 analog input.

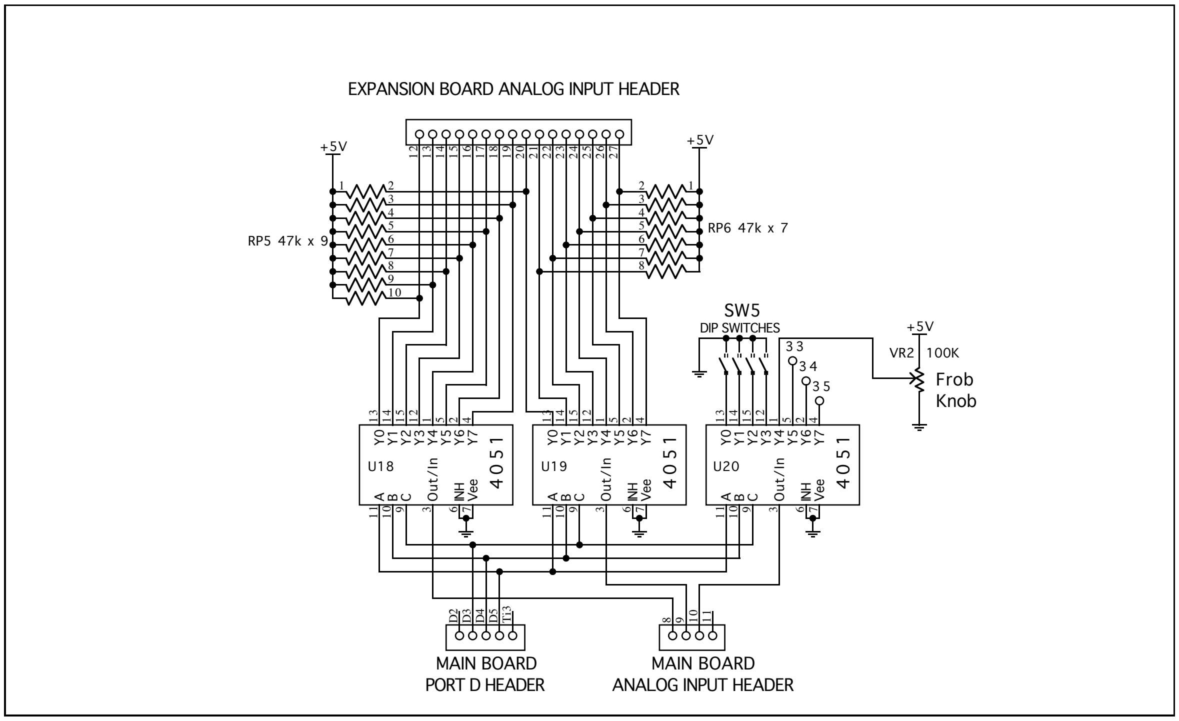

Figure A.16: Expansion Board Analog Input Circuitry

A.4.2 Analog Input Multiplexing on the Expansion Board

The Expansion Board has three *eight-to-one analog multiplexer* ICs. These chips (the 74HC4051) have eight inputs and one output; depending on the state of three selector inputs, one of the eight input lines is connected to the output.[2]

The outputs of the '4051 chips are wired into the 6811 analog inputs when the 6.270 Expansion Board plugs into the main board. Three signals from the 6811 are used to control the multiplexers and select which analog input is mapped to the 6811 analog input[3].

Figure A.16 is a schematic of the analog input circuitry on the 6.270 Expansion Board. It is easy to see how the use of the analog multiplexer chips greatly expands the analog input capability of the 6.270 hardware:

- Two of the '4051 chips have their inputs wired to a bank of sixteen open sensor inputs.

[2]Actually, the chip's signals are bidirectional, but for the purpose of this discussion, it is convenient to think of the chip as having eight inputs and one output.

[3]These signals are taken from the 6811's *High Speed Serial Port*, a special sub-system of the 6811 that allows it to communicate at high speeds with other 6811's. In the 6.270 application, this functionality is not needed; instead, the signals are used as simple digital outputs.

- The other chip is wired from the *Frob Knob*, a general-purpose analog input knob, and four DIP switches (for user configuration input).

- Three of the inputs to this third chip are open, as is one of the 6811's analog inputs.

A.5 The Serial Line Circuit

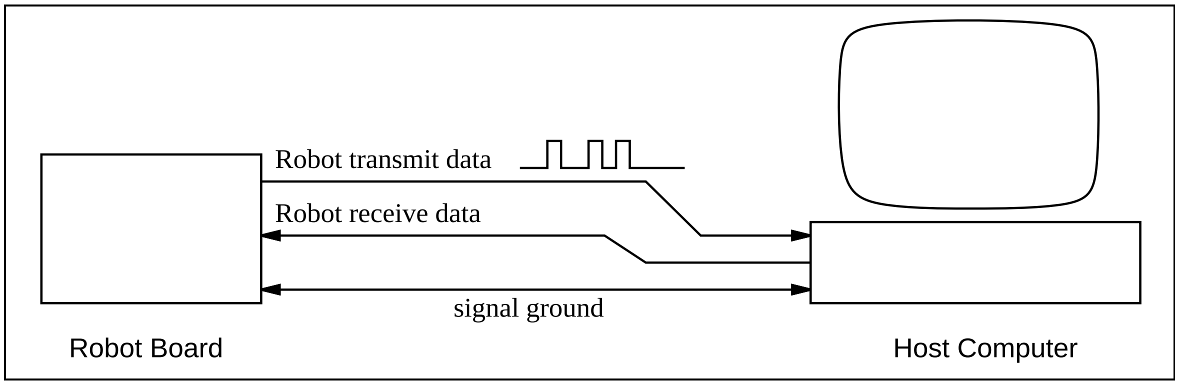

Figure A.17: Host and Board Communications over 3-Wire Serial Link

The 6.270 Board communicates with a host computer over an RS-232 serial line. "RS-232" refers to a standard protocol for communications over a three-wire system, as depicted in Figure A.17. Nearly all of today's computers have serial ports that conform to the RS-232 standard.[4]

In the RS-232 system, a "logic zero" is indicated by a +15 volt signal with respect to ground, and a "logic one" is indicated by a −15 volt signal. Note that this is different from standard digital logic levels in several ways. Negative voltages are used, higher voltages are used, and negative voltages connote a logic one value.

The 6811 chip includes circuitry to generate waveforms compatible with the RS-232 systems, but requires external circuitry to convert its own signals, which obey the digital logic norms, to RS-232 signals as described.

There exist off-the-shelf single-chip solutions to this problem (most notably, the MAX232 and MAX233 chips made by Maxim, Inc.), but these chips are typically expensive and consume a fair bit of power. The solution implemented on the 6.270 board requires a few more components, but is significantly cheaper and less power-hungry.

[4]The actual RS-232 standard involves quite a few more wires for conveying various status information, but the data itself is transmitted on two uni-directional wires.

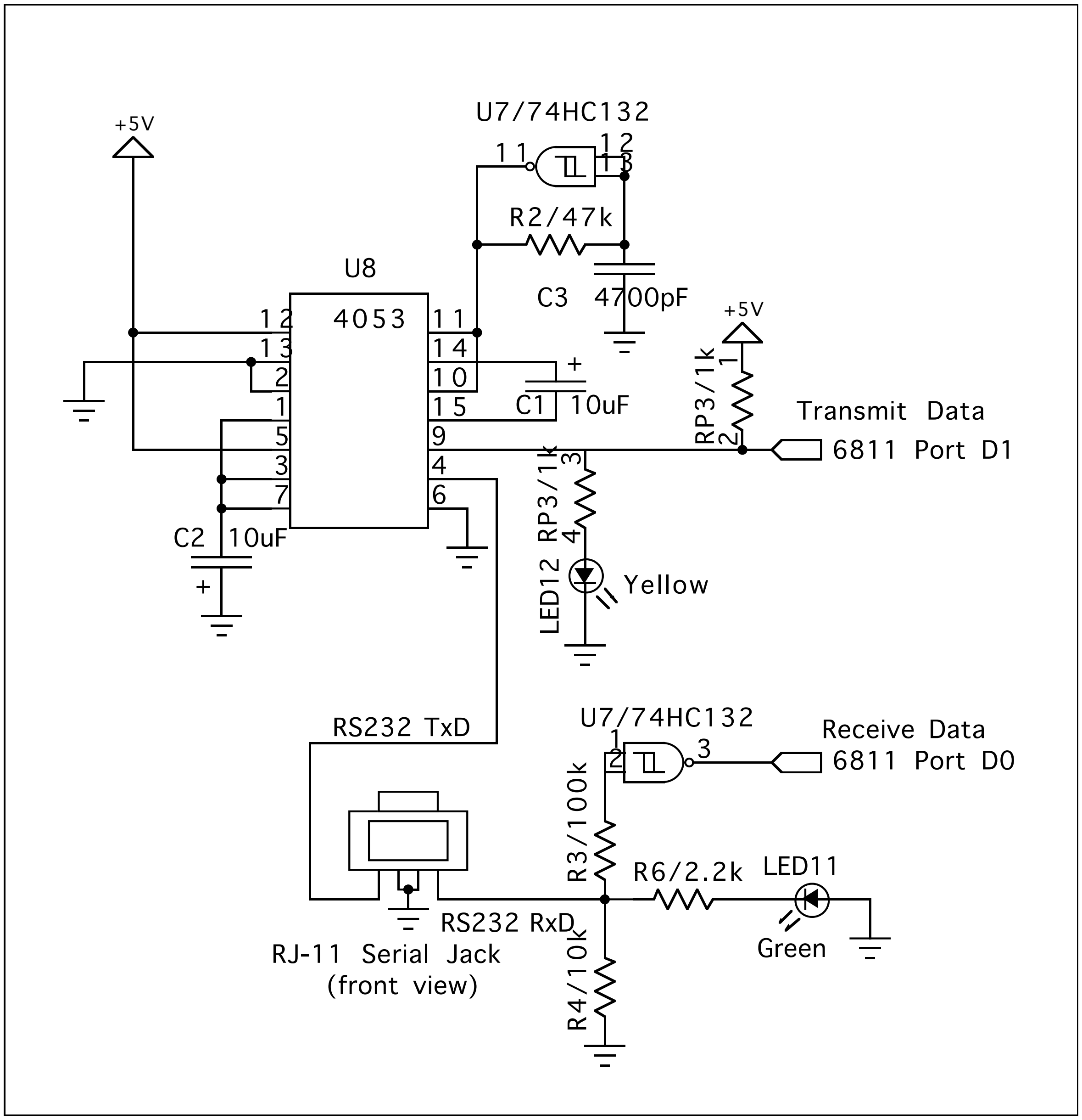

Figure A.18: Serial Line Circuit

A.5.1 Serial Output

One of the difficulties in generating RS-232 signals is obtaining the negative voltage required to transmit a logic one. However, it turns out that the specified -15 volts is not required: -5 volts will do for most applications.

A circuit called a *charge pump* is used to generate this negative voltage. A charge pump consists of two capacitors and a switch. One of the capacitors is charged to a positive voltage by the main power supply. Then the terminals of this capacitor are switched to the terminals of the second capacitor. The first capacitor discharges rapidly into the second, charging it negatively with respect to system ground. This process is switched rapidly, and a steady negative voltage supply is produced in the second capacitor.

The schematic for this circuit and the rest of the serial line circuitry is shown in Figure A.18. The heart of the circuit is a 74HC4053 chip, which is a triple analog SPDT switch that can be controlled digitally.

The charge pump is built from switches A and B of the '4053 chip. Capacitor C1 is charged from system voltage when the switches are in the X position (as is illustrated in the diagram). When the switches are flipped to the Y position, C1 discharges into capacitor C2, creating a negative voltage on C2 with respect to system ground.

The C switch is used to switch either the -5 volts from C2 or $+5$ volts from system power out over the serial line. This is done by wiring the 6811's logic-level "Transmit Data" signal to the control input of switch C.

Switches A and B are repeatedly alternated between the X and Y positions by an oscillator built from a schmitt-trigger NAND gate wired as an inverter (U7) and an RC delay (R2 and C3). This oscillator is tuned to about 10,000 Hertz, a frequency that has been experimentally determined to yield good results.

The commercially-available single-chip solutions mentioned earlier implement a similar circuit. In fact, they use two charge pumps. The first is used to double the system voltage of $+5$ volts to obtain a $+10$ volt supply that more closely matches the RS-232 standard. The second charge pump inverts this $+10$ volts to obtain a -10 volt supply.

A.5.2 Serial Input

A schmitt-trigger NAND gate is wired as an inverter to convert the negative-true RS-232 standard to the positive-true logic level serial standard. Resistor R3 limits the current that can flow into the gate when the serial line voltage is negative, preventing the possibility of damage from a high negative voltage.

The RS-232 standard dictates that a serial line should be in the logic true (negative voltage) state when it is not transmitting data. LED11, the serial receive indicator, is wired such that it will light in this state, being powered directly by the serial voltage generated by the host computer. This LED serves as an indicator that the 6.270 board is properly hooked up to the host.

A.6 Battery-Backing the Static RAM

The static RAM used in the 6.270 board is a special low power device, a relatively recent innovation in widely-available memory technology. This memory chip requires only an infinitesmal amount of current to store its contents when it is not being used.

The actual amount of current—less than one *micro*ampere—is so small that a standard alkaline battery does not notice it. That is, the battery will last as long as its shelf life, whether or not it is supplying one microamp to a circuit. (Alkaline batteries have a shelf life of several years.)

Having a battery-backed static memory greatly increases the usability of the 6.270 board. A robot can simply be turned on and operated immediately, without having to be connected to a computer first.

Unfortunately, implementing a battery-backed RAM can be complicated. The difficulty arises from unpredictabilities in microprocessor behavior when system power is either switched on or off. During these transition periods, the microprocessor is powered by illegal voltages, and its behavior is not defined. In order to make sure that the microprocessor does not corrupt the contents of the memory, orderly transitions from the powered-off to power-on states, and vice-versa, must be implemented.

A.6.1 Powering the Memory Chip

Figure A.14 illustrates how power is alway provided to the memory (through diode D2) even when microprocessor and motor power is turned off. Capacitors C5 and C8 help to smooth the power supply of the memory, and also can provide power to the memory while batteries are being changed. Because the current draw is so small, capacitor C5 will actually keep the memory "alive" for periods of up to thirty minutes when the system is powered off and batteries are removed.

A.6.2 The Power-Off Interrupt

Diodes D1, D2, and D3 provide isolation amongst the three parts of the circuit:

- the memory's power supply
- the microprocessor's power supply
- the power-off interrupt circuit

This isolation is necessary to ensure clean transition of the power-off interrupt circuit when power is shut off. The power-smoothing capacitors (both for the memory and for the microprocessor circuit) retain charge for a brief period after power is switched off. The diodes prevent this charge from "flowing backward," and allowing one part of the circuit to power another.

When power is switched off, the power-off interrupt signal *immediately* goes low. However, system capacitors (mostly, C13) will keep the microprocessor powered up for a short while (about about one-tenth of a second).

The interrupt signal generates a hardware-level interrupt to the 6811. A special-purpose software driver is activated, which has the job of shutting down the 6811 in an orderly fashion before the capacitor power supply runs down.

Sometimes, a brief physical jolt to the microprocessor board will dislodge a battery momentarily, causing the interrupt to be triggered. It would be incorrect for the software to shut down the system in this case. So, the interrupt software waits for a short while to see if the interrupt line goes high (indicating that power has returned). If power does return, the interrupt exits without taking action.

If power does not return after about one-hundredth of a second, the software routine executes a machine-language HALT instruction, which shuts off the microprocessor. This sequence of actions implements an orderly shutdown sequence.

A.6.3 The Power-Up Delays

The Dallas Reset Chip (U11) holds the reset line low for 350 ms after the logic power reaches a level of at least 4.25 Volts. This prevents the 6811 from trying to operate with indeterminate voltages at its inputs, and it safeguards the SRAM while power levels settle.

Once power has normalized, the Dallas chip allows the reset line to rise. The MODA pin has reached a valid logic 1, and the microprocessor comes up in the "run" state. This circuit is shown in Figure A.19.

A second mode is used to download the operating system software to the microprocessor when initializing the board. To bring the processor up in this mode, the MODA must be a logic zero when the processor comes out of the reset state. This happens when the choose button is depressed while the reset occurs. This will put the 6811 into a bootstrap download mode in which a program is executed from internal ROM rather than external RAM.

Diode D3 allows the choose button to pull MODA low without forcing MODA to follow the state of the choose button at all times. Resistor R10 pulls MODA to high when nothing else is going on, so normal run mode is the default after a reset.

If the user presses reset without the choose button, the Dallas chip will pull the reset line low and MODA will remain high. The 6811 will go into the normal run mode, executing a program in external memory.

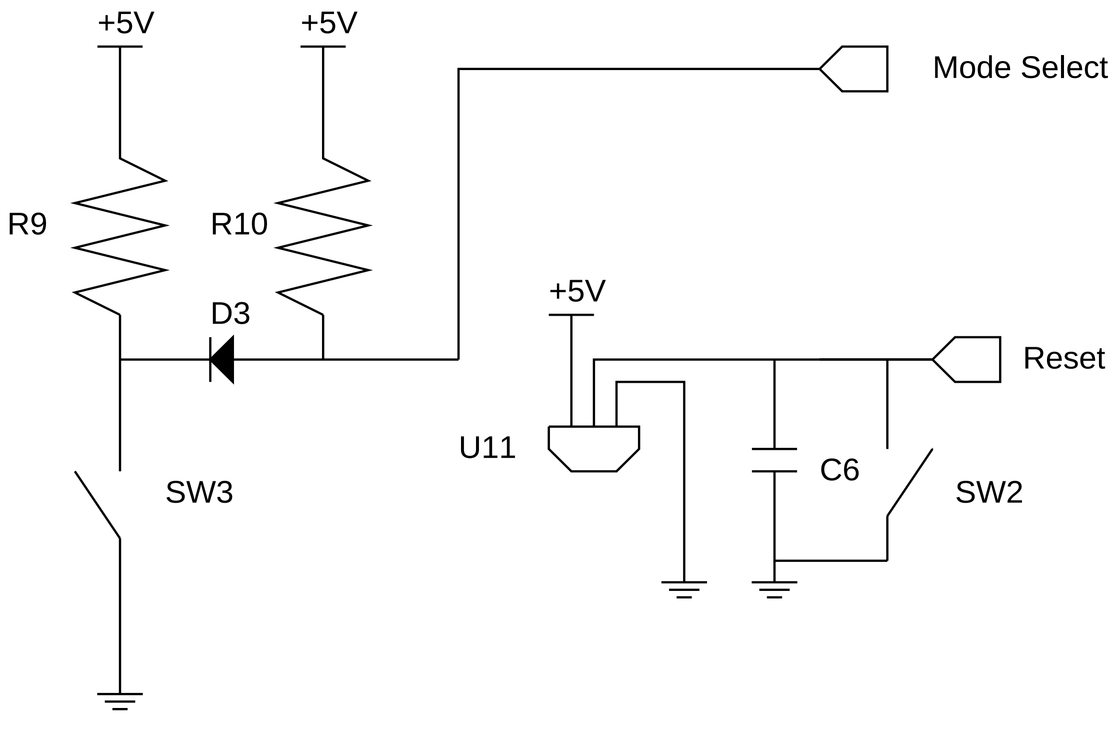

Figure A.19: Reset Circuitry

In order to ensure that the 6811 does not access external memory until the reset conditon is clear, the reset line is also an input to the RAM enable logic.

A.7 The Infrared Transmission Circuit

The Sharp GP1U52 sensor, and others like it commonly used in TVs, VCRs, and other devices controlled by infrared, is sensitive to *modulated* infrared light. It detects the presence of infrared light that is blinking on and off at a particular rate. The GP1U52 sensor is tuned to 40,000 Hertz (40 KHz).

In TV remote applications, a data stream is then generated around the 40 KHz carrier frequency. The signal consists of bursts and gaps of the 40 KHz transmissions.

For the 6.270 application, the 40 KHz carrier is used to tranmit a square wave of relatively low frequency (100 or 125 Hz), as shown in Figure A.20. When the Sharp IR sensor decodes this signal, it removes the 40 KHz carrier, yielding a copy of the squave wave that was originally transmitted (Figure A.21).

Software can continuously check the Sharp sensors for square waves of the specified frequency. It can lock on to the square wave when it is present and count the number of consecutive cycles that have been detected.

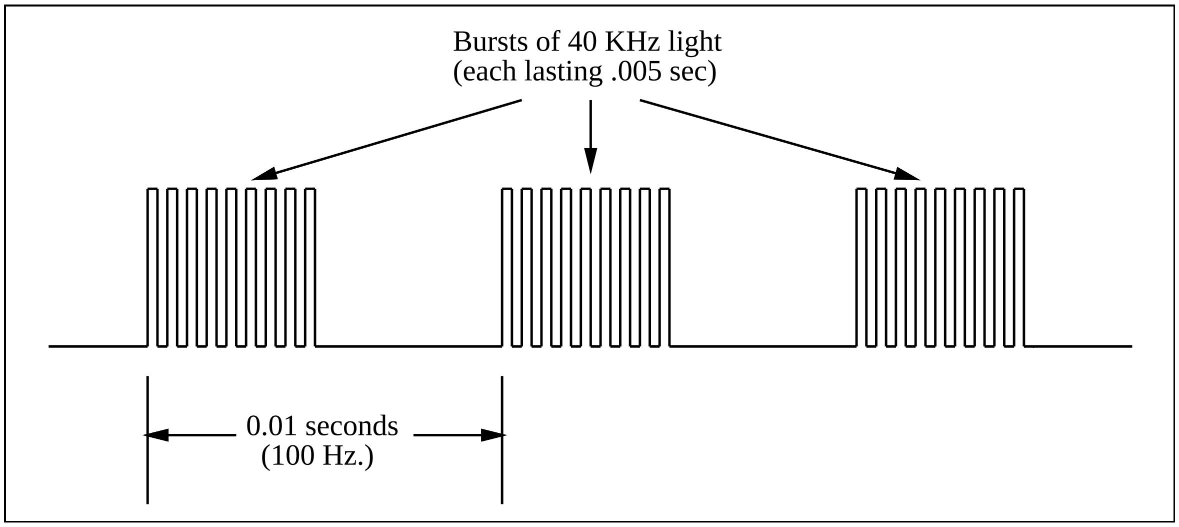

Figure A.20: Square Wave Consisting of Bursts of 40 Khz Signals

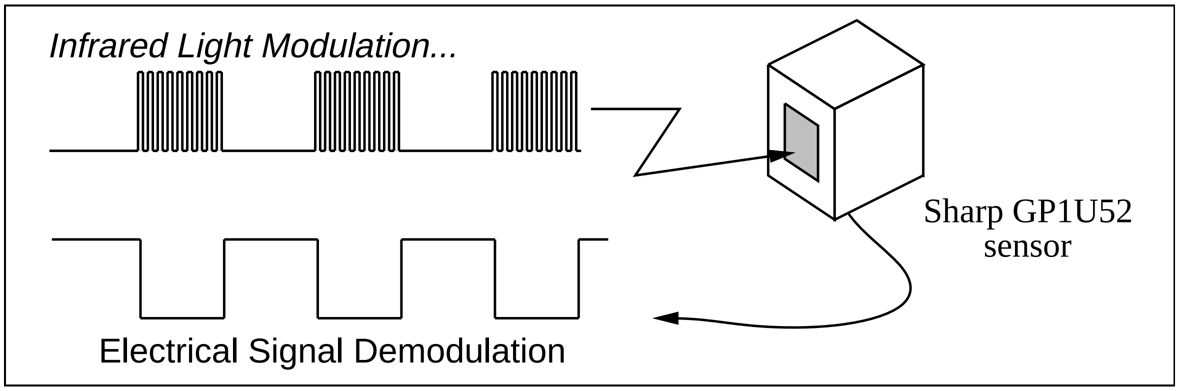

Figure A.21: Sharp IR Sensor Decoding IR-Encoded Square Wave

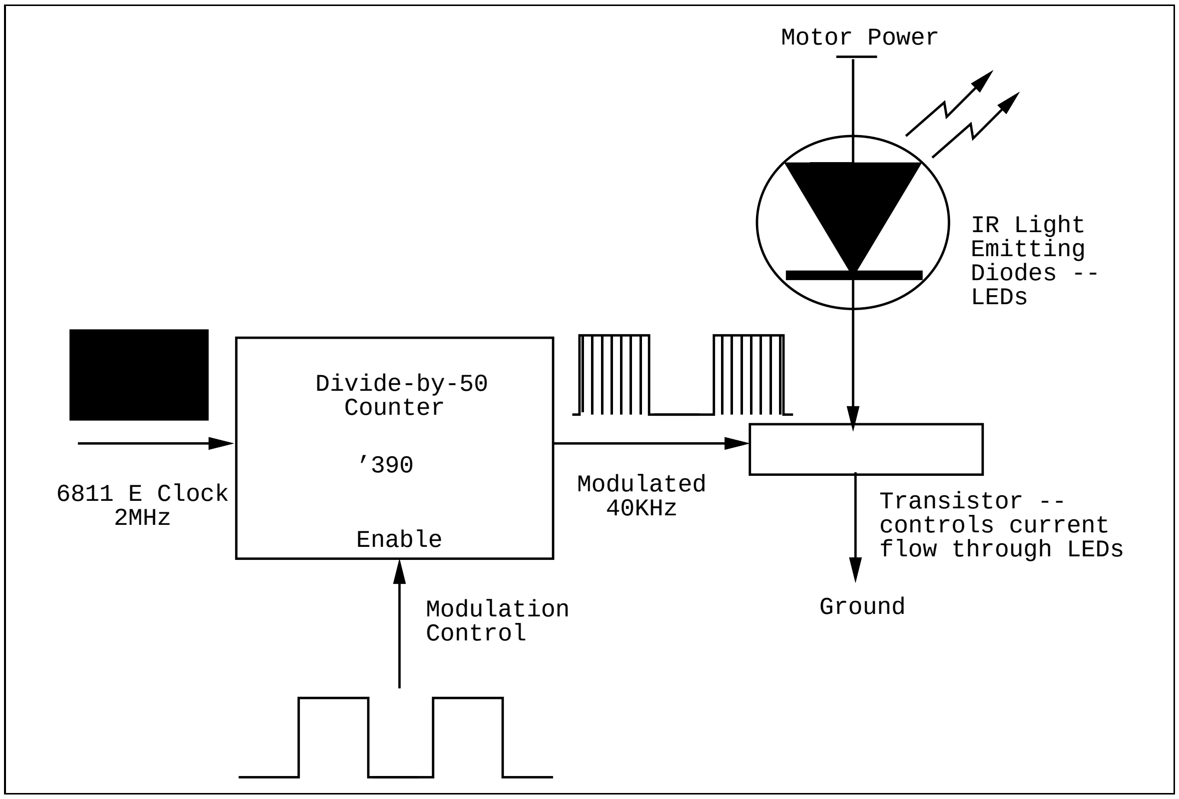

Figure A.22: Block Diagram of Infrared Circuitry

A special circuit is used to generate infrared emissions modulated at the 40 KHz frequency. A block diagram of this circuit is shown in Figure A.22.

The diagram shows that the '390 chip, wired in a divide-by-fifty configuration, is used to generate a 40 Khz signal from the 6811 E clock, a 2 Mhz signal. In actuality, the '390 chip contains two *decade counters*. Each these consists of a separate divide-by-five counter and a flip-flop (a divide-by-two device). The '390 is wired in the divide-by-fifty function by ganging two of the divide-by-five counters and one of the flip-flops.

The *IR control* signal is wired to the clear input of the '390 chip; when this signal is low, the counters will reset and will be prevented from counting. By modulating this signal, the 6811 can generate the low-frequency square wave that ends up being transmitted to the Sharp sensor.

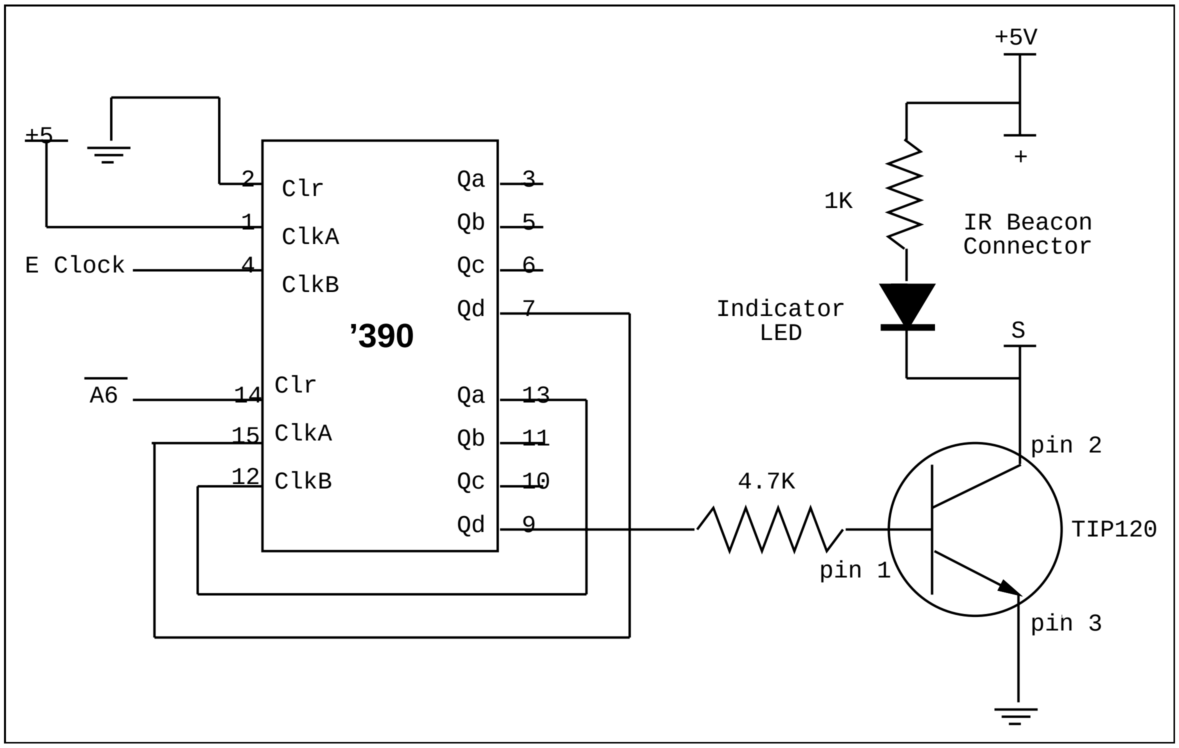

Figure A.23: Infrared Transmission Circuit

Figure A.23 shows the full circuit schematic for the IR subsystem.

The TIP120, a power transistor, is used to drive the infrared LED's. The output of the '390 chip is driven into the base of the TIP120. When this signal is high there is a positive differential between the base and the emitter of the TIP120 and current is allowed to flow from the collector to the emitter, thus driving current through the IR emitter. When this signal is low, the base and emitter are at the same differential and no current flows. This causes no current to be allowed to flow from the collector to the emitter and the IR LEDs have no current flowing through them so they turn off.

A.7.1 The IR Beacon

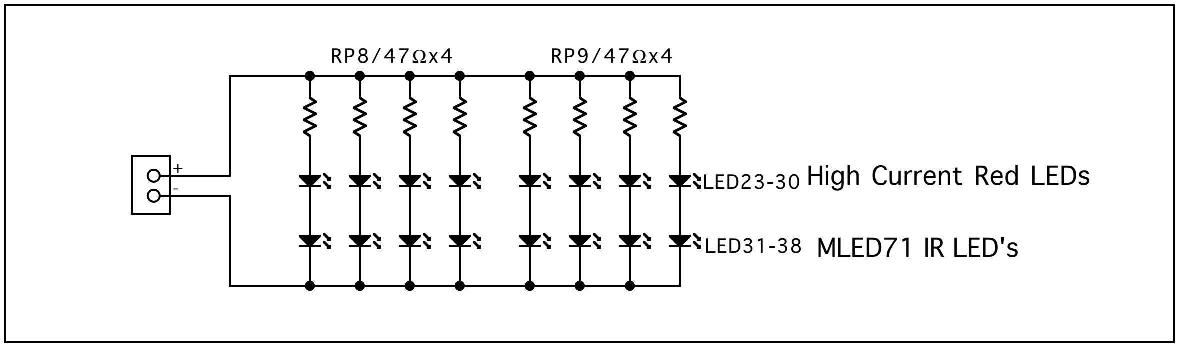

Figure A.24: Infrared Beacon Circuit

Figure A.24 shows the schematic for the IR beacon. Each infrared LED has a visible LED in series with it so it should be easy to ascertain that the device is transmitting infrared light properly. The resistors act as current-limiters, limiting the amount of current that can travel through any branch of the circuit to between 10 to 20 mA.

A.8 The LCD Display

The first fourteen pins of the 6.270 Board's Expansion Bus are designed to be compatible with a 14-pin standard LCD bus. A variety of character-based LCD devices with different screen sizes use this standard bus.

The LCD bus standard is fairly simple, consisting of the following signals:

- an 8-bit data bidirectional bus
- two mode select input signals
- a clock line
- a voltage reference for contrast adjustment
- +5 volt logic power
- signal ground

In fact, reading and writing data to an LCD is much like reading and writing data to latches or to memory. There is one problem, however: LCDs only work at data transfer rates up to 1 MHz. The 6811 in the 6.270 board operates at 2 MHz—too fast for most LCDs.

One straight-forward solution to the speed problem would be to use a '374-type latch between the 6811 and the LCD. The '374 could be written to at the full bus rate of the 6811; its outputs would drive the data bus of the LCD. A separate signal could be used to toggle the LCD's clock line, causing it to latch the data that had been written to the '374[5].

An unconventional, zero-additional-hardware solution has been implemented in the 6.270 system, which takes advantage of an obscure feature of the 6811 microprocessor.

The 6811 has two main operating modes, known as *single chip mode* and *expanded multiplexed mode*. The discussion of memory read and write cycles that has been presented in this chapter has been based on the expanded multiplexed mode, which is the 6811 mode that is used when external memory is part of the 6811 circuit.

When the 6811 is operated in single-chip mode, the upper-eight-bit address bus and multiplexed address/data bus become general purpose inputs and outputs of the 6811, controllable by system software. Thus, in single-chip mode, the 6811 could communicate with the LCD with a software driver, rather than the too-fast hardware communication.

There is a problem with this, however: when the 6811 is placed into single-chip mode, it can no longer execute a program from its external RAM. In fact, as far as the 6811 is concerned, there *is no* external memory anymore.

Fortunately, the 6811 has 256 bytes of internal RAM, from which it can execute a program when in single-chip mode. Thus, a software driver could execute out of internal RAM, perform a transaction with the LCD, and then switch back to expanded-multiplexed mode and return control to the main program in external memory.

The obscure feature mentioned is not the fact that the 6811 has both of these modes, but the idea of dynamically switching between them. Here is the solution that has been implemented:

1. Start by copying a software driver from external system memory into the 256 bytes of internal 6811 memory.

2. Begin execution of the driver program located in internal memory:

    - Place the 6811 into single-chip mode; external memory disappears.

[5]This solution assumes that one does not need to read status data back from the LCD.

- Execute a low-speed transaction with the LCD by directly controlling the data bus via software.
- Place the 6811 into expanded-multiplexed mode.
- Return to the main program in external memory.

3. Continue normal program execution.

The actual LCD driver routine buffers characters to be printed to the LCD; one thousand times per second, an interrupt routine calls the internal memory driver as described, writing a single character to the LCD. The whole process operates transparently to the 6.270 system user.

A.9 The Low-Battery Indicator

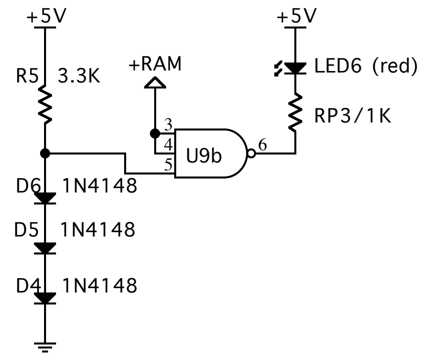

Figure A.25: Low Battery Indicator Circuit

A spare gate on U9 has been used to implement a low-battery indicator. The schematic is shown in Figure A.25.

The transition point for determining if a digital input is logic one or logic zero is normally one-half of the supply voltage. Assuming a 5 volt supply, signals greater than 2.5 volts will be interpreted as logic ones, and signals less than 2.5 volts will be interpreted as logic zeros.

Diodes have the interesting property that they drop exactly 0.6 volts when current travels through them. Thus the input voltage to the gate U9? will be about 1.8 volts, over a wide range of system supply voltages.

Assuming a 5 volt supply, this input would to be interpreted as logic zero. U9? is wired as an inverter, so it will output a logic one. Since the LED is wired from supply voltage, it will be off in this state.

Suppose supply voltage falls to 3.5 volts. Now the transition point is around 1.75 volts. The input to the gate is 1.8 volts, so it becomes a logic one. U9? inverts this to obtain a logic zero, and drives zero volts on its output, lighting the LED.

The actual transition point in the circuit is closer to 4 volts, because the diodes tend to drop a bit more than 0.6 volts that are usually specified. Surprisingly, nearly all of the 6.270 electronics, including the 6811 microprocessor, work fine at voltages as low as 4 volts. One notably exception is the Sharp GP1U52 sensor: its performance decreases sharply at supply voltages less than 4.5 volts.

A.10 Fun Hacks

Many people have created clever hardware hacks to the 6.270 board. Hopefully this section will grow as more and more features are found to further develop the board.

A.10.1 Adding a Loudspeaker

There are two rather simple ways to add an external loudspeaker to the 6.270 board. Teams have done this to play music loudly. However, it requires some minor hardware modifications, in particular, the loss of at least one unidirectional motor port on the Expansion Board. The first version of this hack uses both sides of Motor 5. The second version uses only the right unidirectional side of Motor 5.

Speaker Hack, v1.0

Steps for Speaker Hack v1.0:

- ○ Find Component Side of Expansion Board.
- ○ Find trace under the 74HC374 socket, from pin 19 of the '374 to pin 15 of the L293D. This trace extends underneath the '374 socket left toward the VR2 pot, then up through the false LCD Connector. There is an unused hole, immediately left and slightly above pin 20 of the '374 socket that this trace runs through.
- ○ Cut the trace between the unused hole and the '374 socket with a sharp knife.
- ○ Place a single Female Header pin into the unused hole. This pin should now connect with pin 15 of the L293D.
- ○ Run a wire from the + signal of the Piezo on the Controller Board into the Female Header you placed on the expansion board. There are several unused holes on the Conroller Board for the piezo that you can solder to. Use Male Header on this signal wire so you can disconnect the Expansion Board from the Controller Board.

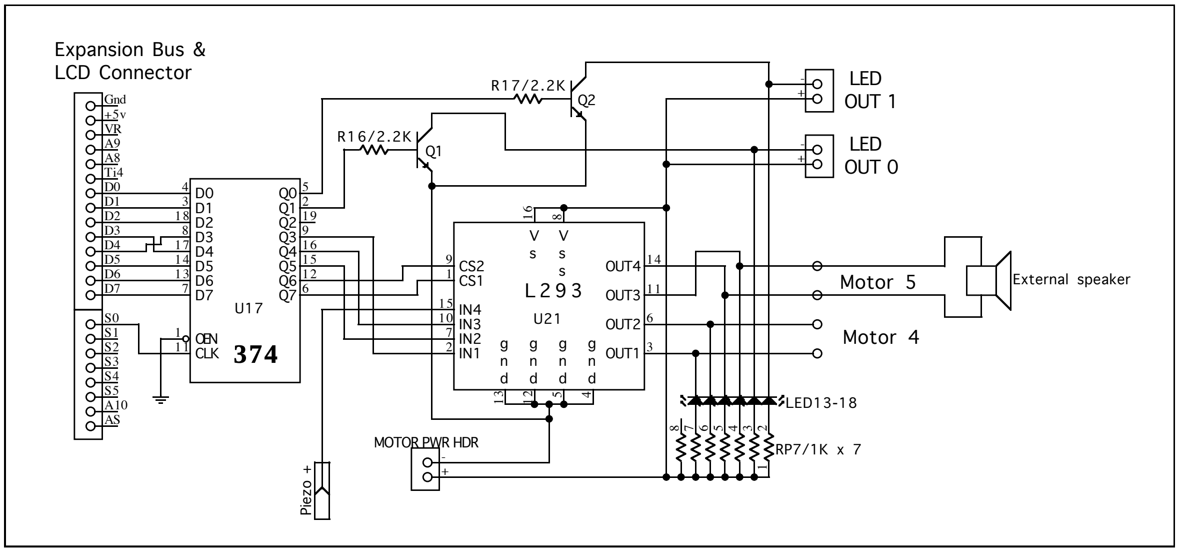

Figure A.26: Speaker Hack, v1.0

◯ Plug your 8Ω external speaker into the bidirectional Motor 5 port. You may wish to put a 100Ω-200Ω pot in series with the 8Ω speaker as a volume control.

◯ In IC, use the command:

```
fd(5);
```

to turn on the speaker. Use the command:

```
off(5);
```

to turn off the speaker.

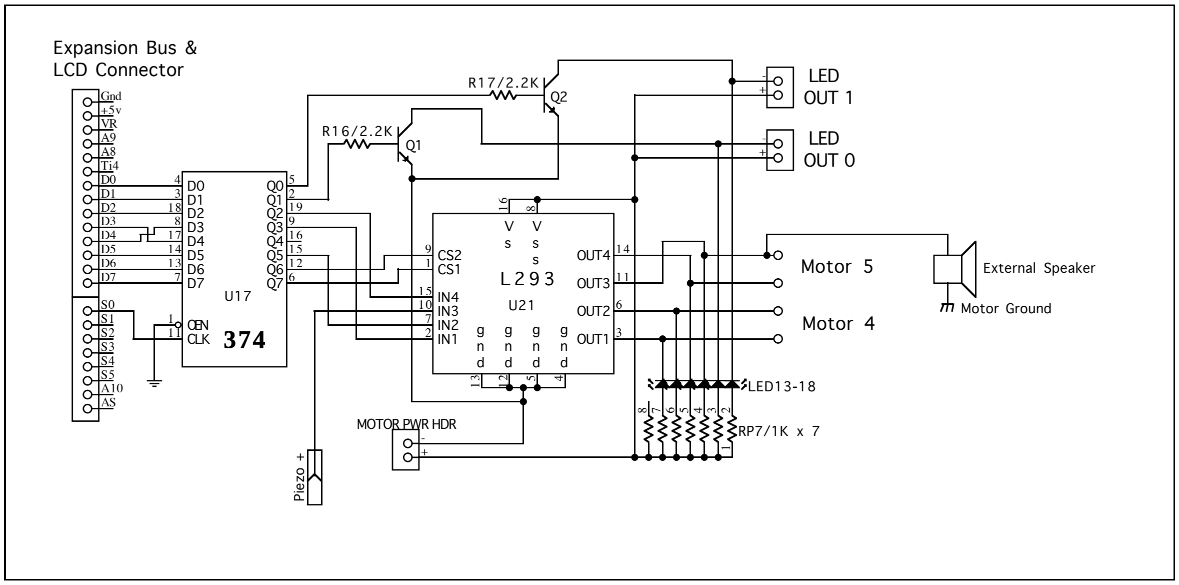

Figure A.27: Speaker Hack, v2.0

Speaker Hack, v2.0

Steps for Speaker Hack v2.0

- ◯ Find Component Side of Expansion Board.
- ◯ Cut the trace from pin 16 of the '374 to pin 10 of the L293D. This trace should be cut immediately above the '374 socket at pin 16.
- ◯ Find the column of holes on the bottom right corner of the prototyping area, immediately to the left of the U19 label. Look one column left of that, directly above pin 10 of the L293D. Solder a single Female Header pin there.
- ◯ Make a solder bridge between the Female Header pin you just placed and pin 10 of the L293D immediately below it.
- ◯ Run a wire from the + signal of the Piezo on the Controller Board into the Female Header you placed on the expansion board. There are several unused holes on the Conroller Board for the piezo that you can solder to. Use Male Header on this signal wire so you can disconnect the Expansion Board from the Controller Board.
- ◯ Plug your 8Ω external speaker into the bidirectional Motor 5 port. You may wish to put a 100Ω-200Ω pot in series with the 8Ω speaker as a volume control.

  The right unidirectional motor port of Motor 5 is now wired to drive the speaker. The left unidirectional motor port of Motor 5 may still be used as normal.

◯ In IC, use the command:

```
motor5_right(1);
```

to turn on the speaker. Use the command:

```
motor_right(0);
```

to turn off the speaker.