

eebot

Technical Description

Peter Hiscocks

Department of Electrical and Computer Engineering
Ryerson Polytechnic University

phiscock@ee.ryerson.ca

August 20, 2002

Contents

1	Introduction	2
2	Technical Overview	3
3	Robot Systems	3
3.1	Robot Mainframe Power Supply	3
3.2	Bumper Switches and Knob	5
3.3	Motor Drive	6
3.4	Motor Speed and Direction	6
3.5	Wheel Rotation Detectors	6
3.6	MPU Alive	7
3.7	Guider Sensors	8
3.7.1	Guider Transient Response	9
3.7.2	Line Tracker	10
3.7.3	Pattern Detector	11
4	Signal Summary	11
5	<i>eebot</i> Hardware Test Routines	13
5.1	Motor Controls	13
5.2	Wheel Counters	13
5.3	The <i>ALIVE</i> Indicator	14
5.4	The Battery and Bumper Switches	14
5.5	The Guider	15
5.6	The Frob Knob	15
6	References	16

1 Introduction

eebot is a small mobile robot intended for learning exercises in robotics, artificial intelligence and microprocessor programming. A photograph of the *eebot* prototype is shown in figure 1 on page 2.

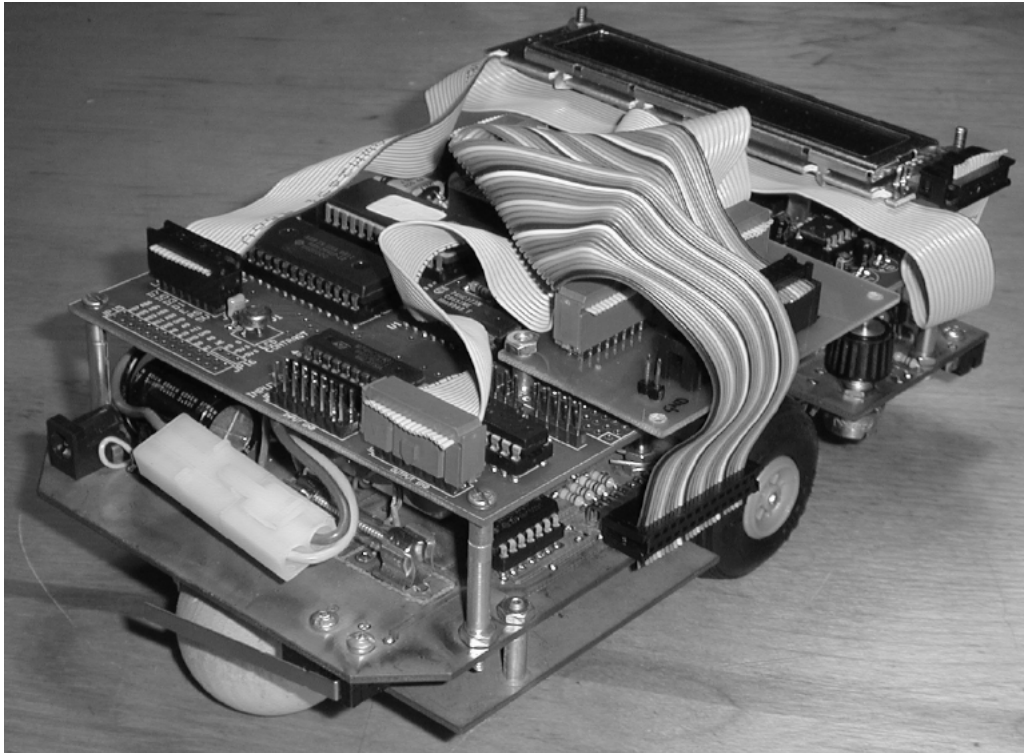


Figure 1: eebot

Many courses that involve mobile robot exercises require students to construct their own robot. This is undoubtedly an educational process but because of time constraints it means that much of the focus is on the mechanical and electrical design and construction of the bot.

In order to focus on programming issues, we have decided to provide students with a 'ready to run' mobile robot. The robot includes a complete microprocessor development system, interfaced to the robot motors and sensors. The robot can be operated from an AC adaptor or from an internal 9.6 volt rechargeable battery.

Programming the bot begins with basic exercises such as reading the sensor array and controlling the speed and direction of two DC drive motors. After this is completed, the student will program the bot to follow a black line, detecting and making decisions at line branches.

Software modules developed in the first phase of the course can be re-used to form the basis of the guidance program in the second phase of the course.

2 Technical Overview

The *eebot* consists of a 3 board sandwich. From top to bottom, the layers are

- A 68HC11 microprocessor development system (the MPP Board, unmodified).
- The robot mainframe
- The sensor board.

In addition, a small printed circuit board is used to adapt the robot mainframe cable to connect to the MPP board.

The MPP Board is a stand-alone development system for the 68HC11 microprocessor. The processor itself includes an A/D converter, input capture interrupts, and output timers, all of which are used as part of the *eebot* system. The development system, external to the microprocessor, includes a monitor program in EPROM, random access memory, an 8 bit input port, an 8 bit output port, and a 20x2 liquid crystal display. The MPP board is available as a kit with assembly manual. It is described in detail in separate publications.

The mainframe contains the robot power supplies, bow and stern bumper switches, DC gearmotor drive motors, power switches, LED indicators, a general purpose potentiometer (the *frob* knob). It supplies DC power to the MPP board and brings signals to and from the MPP board via a ribbon cable.

The sensor board provides optical signals that allow the robot to track a black line and detect branches in the line. It includes 6 high-intensity LEDs to illuminate the line and 6 corresponding Cadmium Sulphide photoresistors to detect the background and line illumination. (CdS photocells are low resistance when illuminated and high resistance when dark.) Four of the photoresistors are configured in a T formation to detect line branches and junctions. Two photoresistors are arranged differentially to track the edges of 3/4 inch wide black electrical tape. The resistances of the cells are converted to voltages and then multiplexed into one channel of the 68HC11 A-D converter. The sensor board also supports the bot front roller.

3 Robot Systems

In the next section, we describe each of the robot subsystems in detail.

3.1 Robot Mainframe Power Supply

For development work on the lab bench, the robot can be powered from a 9 volt DC adaptor. When the program has been developed and is working, the bot can be powered by a 9.6 volt rechargeable NiCad battery, which fits into the robot interior, and then turned loose on the world.

It is advisable to disconnect the battery when using the AC adaptor supply.

Raw 9 volt power is supplied to the MPP board, which then regulates it down to 5 volts to operate the microprocessor.

The 9 volt input power is also regulated down to 5 volts on the robot mainframe, for power to circuitry on the mainframe and sensor boards.

A separate regulator drops the 9 volt input power down to 4 volts or so to operate the drive motors. This is equipped with a large heatsink because the voltage drop (from 9 to 5 volts) and drive currents (maximum about 400mA per motor) combine to generate significant heat in the regulator¹.

¹A switching regulator would be much more efficient of battery power, but might have difficulty coping with the variable current motor load. To simplify the development process, we went with the less efficient linear regulator.

The bot logic is enabled with a LOGIC toggle switch, which is indicated with a POWER LED. A separate MOTOR switch enables motor power.

Logic voltage is connected to one of the A/D channel inputs so that the microprocessor can detect when the battery is reaching the end of its charge. The measurement voltage is about 9 volts, and the maximum voltage readable by the A/D is five volts, so the logic supply is divided down by a factor of 2 for measurement. The software must consequently multiply the A/D reading by 2 to get the correct logic supply voltage.

The logic and the two DC motors share the same 9 volt input power, which can lead to problems of electrical noise. The DC motors tend to pull down the supply voltage when they start, because of their large starting current. As well, the DC motor commutator generates switching noise. To avoid noise contaminating the logic supply

- The motor supply has its own regulator
- Motor current is filtered with an inductor and filter capacitors
- The logic and motor supplies are isolated with rectifier diodes
- The logic power has its own, large reservoir capacitor which can hold up the logic voltage for a few milliseconds

The simplified circuit for the power supply monitor input is shown in figure 2.

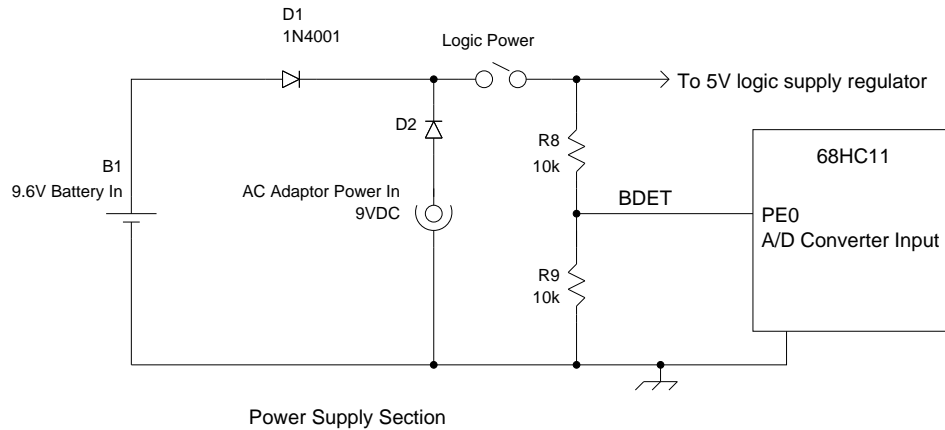


Figure 2: Voltage Monitor, Programming Model

Notice:

- The battery voltage reading must be calculated from the A/D reading. The battery voltage is actually twice the A/D voltage reading, plus one diode drop (0.6 volts). The software routine that displays battery voltage must take this into account.
- When both the battery and AC adaptor are plugged in simultaneously, the effect of the diodes is to select the larger of the two voltages.
- The LOGIC POWER switch must be closed to get a reading.

- The motor is also powered from the same sources, but this is not shown in the programming model. When a motor is running (and especially when it is starting), the logic voltage reading will decrease.

3.2 Bumper Switches and Knob

When the robot bumps into some object, it should stop immediately. Two bumper switches, one at the bow and one at the stern, generate signals for this purpose. When a bumper switch is actuated, the corresponding LED on the back deck will illuminate.

The simplified circuit for bumper switches and General Purpose Knob is very simple, as shown in figure 3.

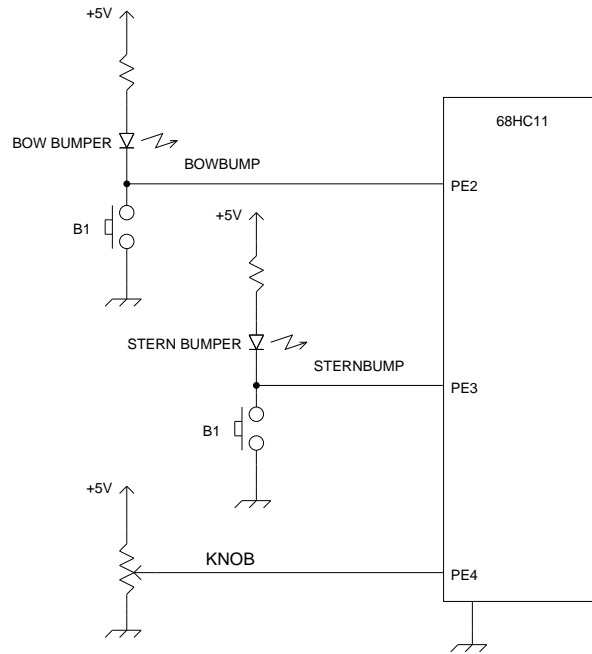


Figure 3: Bumper Switches, Programming Model

Notice:

- When the switch is open (no bump), there will be no current through the resistor and LED. The LED has a fixed voltage drop of about 1.5 volts, so the input voltage to the A/D converter channel will be about 3.5 volts.
- When the switch is closed (bump), the input voltage will be zero volts.
- The general purpose knob (aka *frob* knob) produces an input between zero and 5 volts, which will read between \$00 and \$FF at the A/D input channel PE4.

3.3 Motor Drive

The robot is driven by two DC gearmotors. Each motor is geared down by a factor of 203:1, so the bot has lots of torque for scaling hills, but it moves fairly slowly. It is not intended to be a race car.

To steer the robot, the gearmotors can be driven at different speeds or even in different directions. The robot can then execute very tight turns. The front wheel rolls normally when the robot is moving straight ahead, and skids sideways when the robot is executing a turn.

(An alternative drive system would have been to use *stepping motors* to drive the robot. A stepping motor rotates a certain fixed amount with each pulse of current to the motor coils. This makes it simple to keep track of wheel rotation. However, stepping motors tend to skip steps if their load is exceeded, in which case the distance count becomes erroneous.

DC motors, as used in *inebot*, have more torque than stepping motors, especially when geared down as in this design. However, you then need wheel rotation detectors to detect distance, and you need to be careful about the direction of wheel rotation.)

3.4 Motor Speed and Direction

Each DC motor is operated from a DC voltage of about 4 volts that can be duty cycle modulated. For example, if the duty cycle is such that the supply voltage is on for 50% of the time and off for the remaining time, the average voltage across the motor is then 2 volts, and the motor turns more slowly.

The duty cycle for each motor is controlled from a 68HC11 Output Timer line, which is ideally suited for varying the duty cycle of a pulse signal.

The direction of each motor is controlled by a general purpose computer output signal, so that a logic 0 drives the motor forwards and a logic 1 drives it in reverse.

The motor requires at least 1 volt to operate, the duty cycle range between 25% minimum and 100%. The optimum switching frequency for the motor supply appears to be around 50Hz.

Two LED indicators are attached to each motor, one for each connection to the motor, labelled as MOTOR+ and MOTOR-. These are useful for determining that the motor is receiving power, along with a visible indication of the duty cycle.

The simplified circuit for the motor control circuit is shown in figure 4.

Notice:

- The direction is controlled by a DPDT switch which controls the direction of current flow through the DC motor. The state of this switch is controlled by bits D0 and D1 of the general purpose output register, which is mapped into the microprocessor memory location \$1100.
- The speed of each motor is controlled by a switch in series with the power supply. Varying the duty cycle of the switch sets the average voltage across the motor, and hence its speed. The motor speeds are controlled by the Output Capture lines OC3 and OC4, which are part of the timer section of the 68HC11 microprocessor.

3.5 Wheel Rotation Detectors

For each motor, one of the gearbox gears has holes in it that are detected by a *photointerrupter*² There are 4 holes in the gear and a 13:1 ratio between the gear rotation and the rotation of the output shaft. Counting the holes

²The photointerrupter consists of a light-emitting diode and a phototransistor with a space between them. If something blocks the light path between the two the phototransistor turns OFF. If the optical path is transparent, the phototransistor turns ON. By detecting the state of the phototransistor (ON or OFF), it is possible to sense the passage of a mechanical object through the gap.

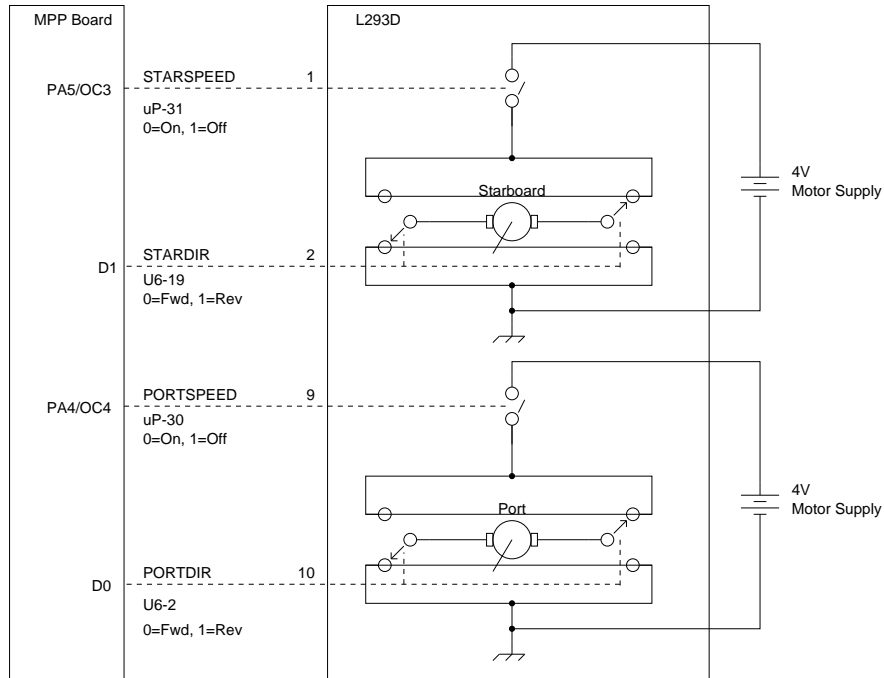


Figure 4: Motor Control, Programming Model

gives a measurement of distance travelled by each drive wheel, with a resolution of about 5mm distance moved per count.

The count information is connected to computer *input capture* inputs so that the computer can monitor wheel count. It can then be used by the computer to ensure that the robot is moving in a straight line or executing a sharp turn through a desired angle.

The photointerrupters are each connected to a ROTATION LED on the rear deck of the mainframe. When the motors are running at slow speed, these LEDs can be seen flashing.

The simplified circuit for the rotation detector circuits is shown in figure 5.

3.6 MPU Alive

When debugging a microprocessor system, it is very useful to have some indication that the computer program is executing correctly. If the computer program crashes for some reason, the indicator shuts off.

This can be accomplished by having the computer pulse an output line every time it executes *the main loop* (also known as *the event loop*). This is a feature that the programmer must add to the system software.

The *alive* circuit on the mainframe detects this type of pulse and causes the MPU ALIVE LED to illuminate when the pulse is present. If the pulse ceases for some reason, regardless of whether the output of the computer is in the high or low state, the LED will extinguish. Think of it as the heartbeat of the computer program.

The ALIVE signal is derived from the microprocessor output signal PA6/OC2. This pin must be configured by the software as a general purpose output pin, which then is flipped in polarity each time the software executes

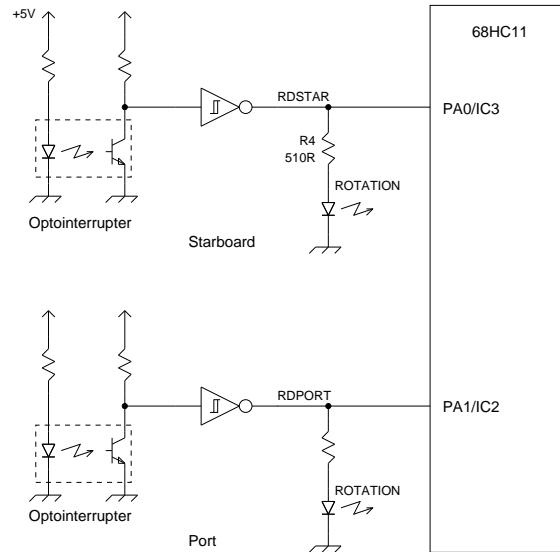


Figure 5: Wheel Rotation Detectors

the main loop of the program.

3.7 Guider Sensors

The Guider Sensors are used for pattern detection and line following of black electrical tape on a light background. (Red electrical tape on a dark background also works well, but of course the *on-tape* and *off-tape* signals are opposite since the contrast is reversed.)

The sensors are 6 Cadmium Sulphide (CdS) photoresistors that are high resistance in darkness and low resistance when illuminated. The sensors are labelled **a** through **f**, as shown in figure 6.

The small circles are high-intensity red LEDs, which illuminate the floor surface under each sensor. The large circles are the sensors themselves. Lines join each LED to its corresponding sensor.

The relationship between a typical sensor and LED is shown in the bottom half of the diagram. The LEDs are mounted on the top side of the guider circuit board and illuminate the floor surface via a hole in the guider circuit board. This arrangement allows the CdS cells to be very close to the floor surface, which improves the sensitivity and reduces the effect of ambient light. As well, the LEDs are protected from being knocked out of alignment.

The simplified schematic for the Guider Sensors is shown in figure 7.

The LEDs and CdS cells are selected by the same signals, the three general purpose digital outputs D2, D3 and D4. The 74HC138 decoder that drives the LEDs is enabled and disabled by the general purpose digital output D5.

One of the voltages developed across the sensors is selected by a 74HC4051 analogue multiplexer and directed to the 68HC11 A/D input channel PE1. The same three general purpose digital outputs D2, D3 and D4 selecting the LED also select the corresponding photosensor. Notice that 3 of the 8 possible sensor inputs are unused and connected to ground, so they should read zero.

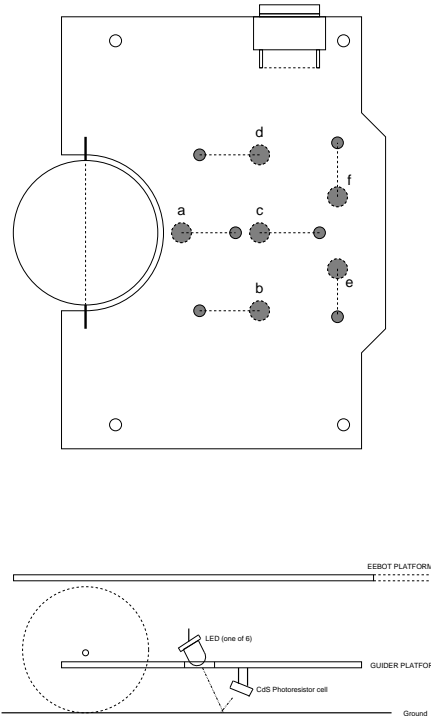


Figure 6: Guider Sensor Locations, Top View

In general, the LEDs will be enabled so D5 should be set to a high (logic 1, +5V) level. The D2, D3 and D4 signals will have some value between 000 and 100 (msb-nsb-lsb) to illuminate one of the LEDs and simultaneously select one of the guider signals. The guider software could scan the 5 sensor signals or repeatedly read one sensor, whatever is required.

To detect the background illumination, the LEDs can be disabled by lowering the D5 signal. The sensor readings would then indicate read the lighting without any LED illumination. This could be used as a baseline or *noise* value and subtracted from the value obtained when the LED is activated.

(The original guider design illuminated all LEDs at once, but that caused problems in light leaking from one LED into several sensors. Furthermore, there was no way to determine background illumination level.)

3.7.1 Guider Transient Response

The CdS photoresistors respond quickly to an increase in illumination (decreasing resistance) but rather slowly to decreasing illumination (increasing resistance). Consequently, for accurate measurements the photoresistor should be allowed time to darken between each measurement. The darkening time constant is in the order of 50 milliseconds³. Consequently, the most rapid allowable scan rate to scan one sensor is 20 times per second. If all five sensors are being scanned, the fastest allowable scan rate increases because a given sensor can be darkening

³This was determined by aiming an LED directly at a photoresistor and pulsing it from a function generator while measuring the voltage across the photoresistor with an oscilloscope.

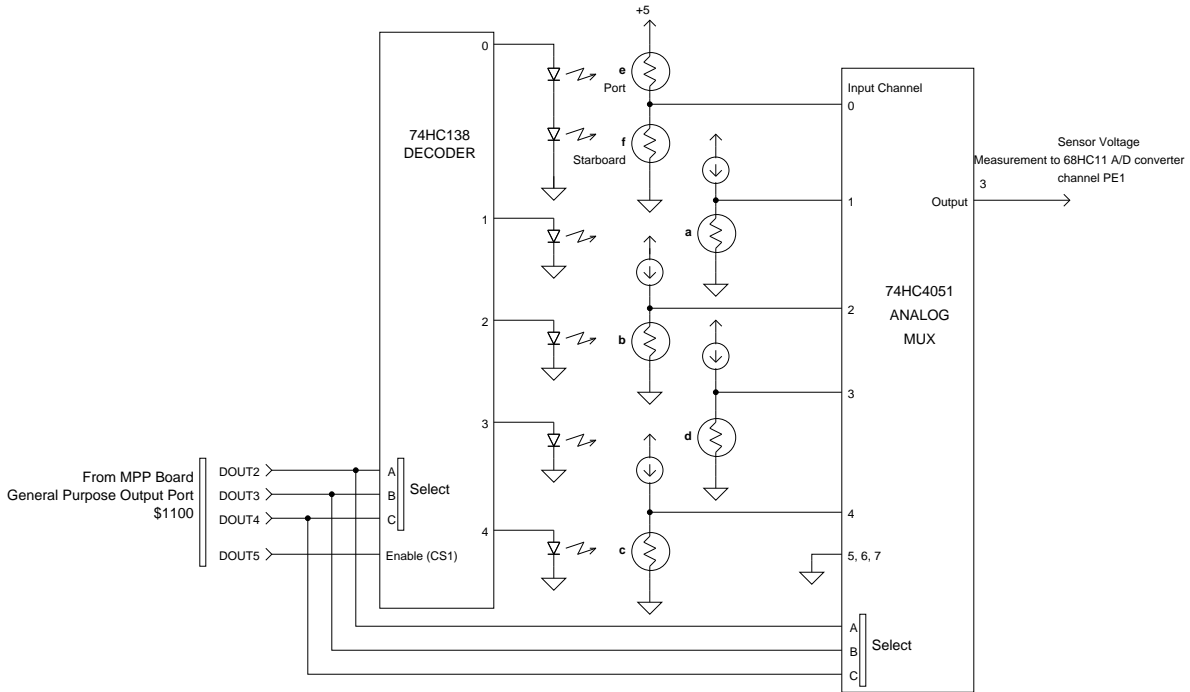


Figure 7: Guider Sensors

while others are being read. Then the minimum scan time per sensor is 10 milliseconds, or about 100 times per second.

The differential line tracker pair of cells **ab** actually recovers faster than a single cell, and the effective time constant is about 25 milliseconds. (This is because one of the pair is brightening while the other darkens.) So the line tracker signal could be scanned as fast as 40 times per second, which may be important for rapid response in steering.

3.7.2 Line Tracker

Sensors **e** and **f** used for *line tracking*. They are spaced 0.75 inches apart, so that they are centered over the two edges of electrical tape line when the robot is centred over the line. The two sensors are a voltage divider, so that the voltage at the centre tap is determined by the ratio of the two sensor resistances. The guidance software of the robot should position the robot so that the voltage at the centre tap from voltage divider sensors **e** and **f** is exactly midway between 5 volts and zero, ie, 2.5 volts. Then the two sensor resistances are equal, the light level on the two cells is equal, and the two cells are equally over the black line and light background.

Notice that the absolute value of the resistance of sensors **e** and **f** is unimportant: it is their ratio that determines the output voltage. Consequently, they should largely ignore changes in ambient light level and this does indeed prove to be the case.

However, if both sensors are over a completely black background or a completely white background, they will also generate 2.5 volts. Consequently, some other method is required for ensuring that the robot is over a valid

line, and that is the function of the pattern detector.

3.7.3 Pattern Detector

Sensors **a**, **b**, **c** and **d** form a *pattern detector* for various line configurations. For example, if sensors **a** and **c** detect a dark line while sensors **b** and **d** detect a light background, then the robot is probably over a valid line and the information from the line tracker can be used for guidance.

If the pattern detector sensors all detect a light background, then the robot is off line, and should begin to search for a line. The line tracker cannot be used.

If the pattern detector senses a black line at right angles to the current line (for example, sensor **a** reads light, sensor **b** reads dark, sensor **c** reads dark, sensor **d** reads light, so the line angles off to port), then the robot should advance until the drive motors are over the bend in the line and then pivot counter-clockwise until the spur line is under sensors **a** and **c**. Similar interpretation strategies should enable the pattern detector to sense a **T** junction or a line that simply comes to an end.

The sensors of the pattern detector each detect absolute light level. A constant current is passed through each sensor and the voltage generated across the sensor is then proportional to its resistance. The sensor currents are each adjusted so that the sensor generates about 1.8 volts over a light surface and 3 volts over a dark surface⁴.

4 Signal Summary

The *eebot* signals and their connection to the microprocessor system are shown in figure 8 on page 12.

⁴As part of the operating procedure, the operator should check the A/D reading from each of the pattern detector sensors over light and dark surfaces, and calibrate the software thresholds accordingly.

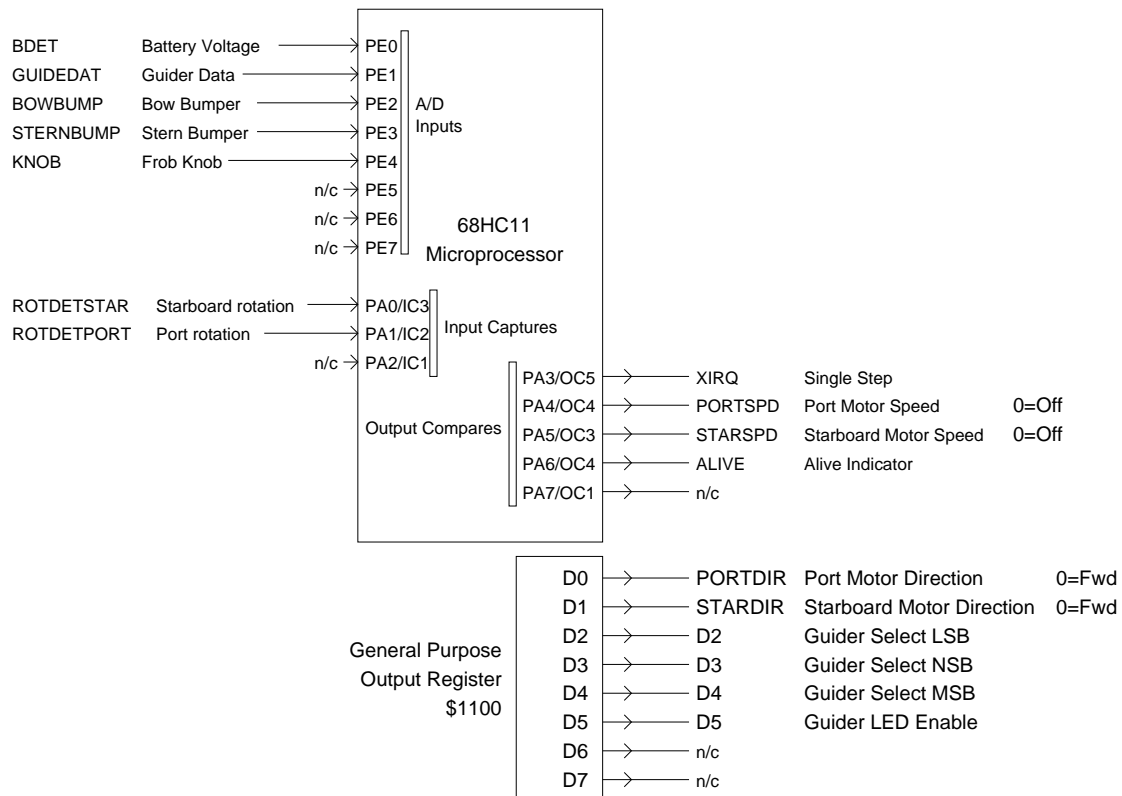


Figure 8: Signal Summary

5 *eebot* Hardware Test Routines

In this section, we show test routines that may be exercised from the microcomputer monitor program. These are routines that may be used very quickly, without programming, to check whether some hardware feature of the *eebot* is working correctly.

All these test routines assume that the robot is powered up correctly and that the computer is connected to a host running a terminal emulator program. Contact with the monitor program has been established via the serial terminal connection on the MPP board.

5.1 Motor Controls

The motors may be turned on and off by changing bits in the 4 and 5 in the PORTA register of the 68HC11 at \$1000:

Monitor Command	Action
mm 1000 xx 00	Turns off both motors
mm 1000 xx 10	Turns on the Port motor (only)
mm 1000 xx 20	Turns on the Starboard motor (only)
mm 1000 xx 30	Turns on both motors

The monitor command mm 1000 xx 00 is interpreted as follows:

- Type in mm 1000 followed by <enter>
- The monitor shows the current value contained in 1000, which is irrelevant and shown as xx above.
- Type in the new value 00, in hex, but without a preceeding \$ sign.

The motor direction may be changed by changing bits 0 and 1 in the GPOUT register at \$1100. (Notice that GPOUT is on the MPP board but not part of the 68HC11, so it's not described in 68HC11 manuals.) A **0** bit selects forward motion, a **1** bit selects reverse.

Monitor Command	Action
mm 1100 xx 00	FWD both motors
mm 1100 xx 01	Starboard FWD, Port REV
mm 1100 xx 02	Starboard REV, Port FWD
mm 1100 xx 03	REV both motors

5.2 Wheel Counters

The technique for reading the wheel counters is to

- Start one of the motors turning as described above in section 5.1
- Repeatedly read the PORTA register at \$1000:
 - enter the command
md 1000 1000
 - repeatedly press the <enter> key (or just hold it down)

- Check to see if the bit corresponding to that wheel counter input is changing.

For example if you run the STARBOARD motor, when you repeatedly display `PORTA` you should see the least significant nibble of `PORTA` changing in such a way that corresponds to changes in the least significant bit. For example, it might fluctuate between `FC` and `FD`, or between `FE` and `FF`.

Similarly, if you run the PORT motor, you should see the least significant nibble of `PORTA` changing in such a way that corresponds to changes in the second least significant bit. For example, the reading might fluctuate between `FD` and `FF`.

5.3 The *ALIVE* Indicator

The *ALIVE* indicator may be tested with a short looping program that continuously flips bit `PA6`.

Using the monitor `ASM` command, type in the following program from the monitor prompt line.

```
asm 6000
ldaa 1000
eora #40
staa 1000
bra 6000
```

Start this program with the command `g 6000`. The *ALIVE* indicator should illuminate continuously. Break out of the program by pressing the reset button on the MPP board computer.

5.4 The Battery and Bumper Switches

The battery voltage and two bumper switches are on A/D channels `PE0`, `PE2` and `PE3` respectively. The first four channels of the A/D, `PE0` through `PE3`, may be displayed continuously with the following sequence of commands⁵:

- Start the A/D converter running continuously by executing the monitor command `mm 1030 xx 30`.
- Display the conversion results with the monitor command `md 1030 1030`.
- This will display the contents of the 16 machine registers starting at location `$1030` through `$103F`. Locations `$1031` through `$1034` contain the input data we are interested in, from A/D channels `PE0` through `PE3`:

Address	Channel	Function
1031	<code>PE0</code>	Battery voltage
1032	<code>PE1</code>	Guider (see section 5.5 below)
1033	<code>PE2</code>	Bow bumper
1034	<code>PE3</code>	Stern bumper

- By repeatedly pressing or holding down the `<enter>` key, you can get this memory dump to repeat. (The monitor repeats the last command you entered.)

⁵Tip o' the hat to Ken Clowes, who suggested this technique

Battery Voltage

The battery voltage value may be determined by converting the hex value on channel PE0 to volts, multiplying by 2, and adding 0.6 volts. Each count amounts approximately to 20mV, so a reading of \$DE would correspond to an input voltage 4.35 volts, which is a battery voltage of 9.3 volts.

Bow Bumper Switch

The Bow Bumper signal is on channel PE2. When the switch is not actuated, the voltage should be some non-zero value corresponding to approximately 3.5 volts. When the switch is actuated, the input voltage should be zero. Use the procedure of section 5.4 above to repeatedly display the readings of channels PE0 through PE3. While repeatedly displaying the contents of PE0 through PE3, press and release the Bow Bumper switch. You should see PE2 change.

Stern Bumper Switch

This is exactly the same procedure as the Bow switch, except that the reading is on channel PE3, stored at location \$1034.

5.5 The Guider

The method of testing a guider is as follows:

- start the A/D converter running in continuous mode, as described in section 5.4
- select the guider sensor by setting bits 2,3 and 4 of the GPOUT register at \$1100 according to the following table. Notice that bit 5 must be a logic 1 to enable the LEDs.

Sensor	Contents of GPOUT Binary	Contents of GPOUT Hex
A (front)	00100000	20
B (port)	00100100	24
C (centre)	00101000	28
D (starboard)	00101100	2C
E-F (line)	00110000	30

- dump the machine registers at \$1030 using the md command
- watch the contents of \$1031 while covering and uncovering the sensor to be checked.
- reading a sensor with bit D5 set to zero will give a background illumination reading, ie, a reading with the corresponding LED disabled.

5.6 The Frob Knob

The Frob Knob may be used to check whether the A/D converter is working properly.

The frob knob is connected to A/D channel PE4, so the command to start the A/D converter is slightly different. Otherwise the procedure is the similar to section 5.4:

- Start the A/D converter running continuously by executing the monitor command
mm 1030 xx 34.
- Display the conversion results with the monitor command
md 1030 1030.
- This will display the contents of the 16 machine registers starting at location \$1030 through \$103F. Locations \$1031 through \$1034 contain the input data we are interested in, this time from A/D channels PE4 through PE7:

Address	Channel	Function
1031	PE4	Frob Knob
1032	PE5	No connection, reads 0
1033	PE6	No connection, reads 0
1034	PE7	No connection, reads 0

- By repeatedly pressing or holding down the <enter> key, you can get this memory dump to repeat. (The monitor repeats the last command you entered.)

Move the Frob Knob while watching the contents of location \$1031. The value should increase as the knob is rotated clockwise.

6 References

M68HC11 Reference Manual

Motorola Document M68HC11RM/AD REV 3, 1991

The authoratative source of information about the 68HC11 microprocessor.

68HC11 Microcontroller, Construction and Technical Manual

Peter Hiscocks, 2001

Technical information on the MPP Board, 68HC11 Microprocessor Development System

Information on programming and interfacing the MPP Board used at Ryerson and elsewhere.

M68HC11: An Introduction, Software and Hardware Interfacing

Han-Way Huang

Delmar Thompson, 2001

A basic text on the 68HC11 microprocessor.

Document Production Notes

These notes were prepared on a computer running the Linux operating system, Red-Hat distribution. The text was edited with the joe editor and typeset using the L^AT_EX typesetting program into the times font.

Diagrams were prepared with the xfig drawing program and incorporated into the text in eps (encapsulated postscript) format using the epsfig command.

Photographs originally in jpeg format were converted to postscript format using the xv image manipulation program and incorporated into the text in eps (encapsulated postscript) format using the epsfig command.

Computer code was formatted with the Unix pr utility and included into the text file using the scriptsize font and verbatim environment.

The resultant typeset document was previewed with xdvi. When completely debugged, it was converted to postscript format using the program dvips then to pdf format using the script ps2pdf.