Computer Controlled Car Project
Final Report

Luke Tranlong
743 N. First Avenue
Hillsboro, OR 97124
(503) 648-3546

Fall 1991

Senior Design Project (Course # 35498)

Advisor: Professor George Nagy

Rensselaer Polytechnic Institute

Abstract

The purpose of this research project was to integrate some prebuilt logic to an M68HC11EVB board which would then control the movements of 1/10 scale model car. The functions that were to be manipulated were forward and reverse movement, steering, distance travelled, and range finding through the use of an ultrasonic sensor. A software program was written which would provide the signals necessary to operate the control logic and handle the feedback. Although most of the goals have been accomplished, the ultrasonic sensor is still nonfunctional and needs further work.

Table of Contents

Introduction

The purpose of my senior design project was to assemble a 1/10 scale model car and integrate it with various control logic that had already been designed. An M68HC11EVB single-board computer was then used to control the external logic. Finally, I wrote the software which accepts inputs as to the direction and distance the car is to travel and then give feedback regarding the car’s position in its environment.

To do accomplish the goals, it was necessary to test and debug the various components before integrating them onto the car. This was where the majority of my time was spent. As a result, the criteria by which the project can be judged on is whether or not the subsystems actually function. Furthermore, the software program can also be used to evaluate my performance on the project.

In this report, I will focus on what the subsystems are and how they function. Then, a discussion on the problems that were encountered and the steps that were taken to try and remedy them will be presented. Finally, the control program will be explained in detail.

Background

There were two main reasons why this project was initially undertaken. First, a Ph.D. candidate working on an automated parallel parking algorithm required a platform on which his theories could be tested. Second, this project gave seniors an opportunity to put their classroom experience towards a design application. I fell into this latter group.

When I started the project, I was under the impression that the various components such as the DC motor and ultrasonic sensor control logic were fully functional. However, as will be discussed, a variety of problems were discovered which led to the project not being completed to the specifications stated in the project proposal.

The functional components of the system can be divided into three subgroups. The first of these is the drive train and steering mechanism. It consists of a DC motor, the servo control, and an optical sensor (odometer). The DC motor is operated by applying -5V to +5V at its input terminals. For the odometer, counters are used to detect the number of gear teeth that pass through the sensor beam. Since there are 44 gear teeth on the gear mounted on the axle of the 6 cm diameter wheel, the travelled distance can be determined. Finally, the direction of the front wheels and orientation of the transducers are controlled by two separate servos. These servos are operated by a pulse train with a period of 20 msec. By varying the duty cycle of the constant period signal, the servos can be made to turn and stabilize.

The second subgroup is the ultrasonic sensor-control board. Theoretically, the operation of this component is very simple. The user should only need to tell the computer to emit an ultrasonic pulse and, some time later, be able to read in the distance an object is away from the transducer. Furthermore, by using two sensors, it should be possible to detect even the angle of orientation of the object in relation to the car, since the distance between the two transducers would be fixed. The hardware that controls the transducers is a combination of logic built by previous students and a prefabricated control board purchased from Polaroid. As of this writing, I have not been able to get this part of the project to function correctly.

The last subgroup of the project is the software program. Through a series of instruction sequences, it was possible to make the car's hardware components function simultaneously. By specifying the direction, distance, and angle of the two servos, the car can be made to move. As already mentioned, feedback can be obtained from the odometer. The program has been designed so that once the ultrasonic ranging system works, it can be easily integrated in the software.

Overview of Project

Below is a more in-depth look at the three subsystems. Information not described below concerning the motor and ultrasonic control boards can be found in Mark Kordon's and Vivek Shrivastava's final reports. Any modifications to their designs will be noted.

Motor Control Board

The motor control board uses Ports B ($1004) and C ($1003) of the EVB. Port B is used for the control lines while Port C is used as a read/write data bus. To engage the DC motor, the DAC (U4)[1] must have valid data at its input terminals. Placing onto Port C a value between $00-$70 means the motor will turn in reverse while values between $90-$FF is for forward movement. This is assuming, of course, that the wires from P3 are hooked up with the correct polarity to the motor. Finally, latching of this data into U1 is required. The latching is accomplished by sending $01 to Port B. Note that values between $70 and $90 will cause the DC motor's input voltage to be 0, effectively disengaging the motor. An alternate way of stopping the motor is by sending $02 to Port B. Module U2, which has $82 hardwired as its output, then becomes the data input for the DAC. The DAC

[1] For all references to specific IC's in the description of the motor control board, please refer to Appendix A.

then controls which transistor (T1 or T2) will be turned on which thus engages the DC motor in the proper direction.

For the steering and ultrasonic servos, a pulse train of 50 Hz with a varying duty-cycle is required to change the servos' orientation. To accomplish this, a MC6840 timer chip (U4) is used.[2] This chip consists of three independent timers which can be programmed for different kinds of outputs. Timer 3 is clocked by the E clock from the EVB. This drives timers 1 and 2 which are set for one-shot output signals of a predetermined duration. This time parameter can be varied, thus changing duty cycle.

The last part on the motor control board is the optical sensor which functions as an odometer for the car. To operate the sensor correctly, counters U9, U12, and U13 must first be cleared by sending $03 to port B. After the motor has been engaged, the counters will count how many gear teeth have passed through the optical sensor. To read the count, bit 0 of address $1002[3] must first be cleared. Then, Port C must be set to read status by writing $00 to memory location $1007 (DDRC). Finally, the values from the counters can be read in one byte at a time by placing $04 for the low byte and $05 for the high byte onto Port B. Note that it is necessary to re-assert bit 0 of address $1002 for correct operation of the rest of the circuit.

[2] Note that the values required to program this chip are different than in Mark Kordon's report.

[3] This important step was not addressed in Mark Kordon's report which caused considerable frustration and loss of time.

Ultrasonic Control Board

The logic for the operation of the ultrasonic sensor actually consists of two boards. One was pre-fabricated by Polaroid and purchased for this project. The second board was designed and built by previous students and will henceforth be known as the ultrasonic control board. The Polaroid board works as follows: VSW is a control signal which initiates the board to emit a square wave of 56 cycles labelled XLG. XLG is then routed through the control board so that it can then be passed back to the Polaroid board as XLG', either unmodified or decreased to just 4 cycles. This is done so that shorter distances can be measured without interference between the emitted and received signals. XLG' then drives the transducer through the analog circuitry on the Polaroid board. Finally, when an echo is received, MFLOG, which is normally high, is cleared.

The ultrasonic control board determines the input of the pre-fabricated Polaroid board and processes the output signals. In theory, the operation of the board is quite simple. When the user requires a distance measurement to be made, the counters U4, U5, U6, and U7[4] should first be cleared. This is done by sending $43 to Port B. Then, a signal to start the sonar is sent by placing $44 onto Port B. This should cause transistor T1 to turn on which effectively pulls VSW high. There is a delay between when VSW goes high and when XLG actually starts

[4] For all references to specific IC's in the description of the ultrasonic control board, please refer to Appendix B.

pulsating. Since the exact time delay is critical when we are dealing with such short time intervals, XLG is actually used to enable the counters rather than the start signal from the EVB.

As mentioned above, XLG can be stepped down from 56 cycles to 4 cycles depending on the distance measurement required. Cutting down the number of cycles is done by toggling U17 which in turn controls whether or not all 56 cycles will be passed back to the Polaroid board. Once MFLOG is received, the counters are disabled and thus, the distance from an object can be determined. Note that the counters are driven by a clock pulse of 1.33 MHz which is important in the distance calculations. For further reference on this subject, see Vivek Shrivastava's report, pages 15-17. Finally, the low and high bytes of the counters can be read from Port C ($1003) by placing $08 and $04 onto port D ($1008) respectively.

Software Program

All of the data requirements presented above are incorporated into a BASIC (BASIC11 V1.55) program that can actually make the car move in a predetermined path. The program flow of execution is as follows:

1. Get next set of data points
2. If distance desired=0, exit program
3. Convert data points from user units to machine units
4. Turn steering wheel and sonar mount
5. Clear odometer counter
6. Engage motor in proper direction
7. Send ultrasonic pulse/Read distance from object[5]
8. Check odometer
9. If distance travelled is less than desired, repeat 6/7
10. Go to 1

A listing of the program is included in Appendix D. The program basically follows the algorithm set forth above. Therefore, only the unusual lines of code will be explained in greater detail.

The data that is expected from the program is as follows: distance to travel (-744 to +744 mm, negative being backwards), degrees to turn the steering wheel (-30° to +30°, negative being to the left with respect to the car), and degrees to turn the ultrasonic sensor mount (-90° to +90°).

In converting from user units (mm, degrees) into machine units (gear teeth, timer counts), several important factors must be considered. These problems are mainly due to the limitations of the 6811 BASIC chip which comes with the EVB. Since only

[5] As of this report dated December 9, 1991, the ultrasonic control board does not function correctly. However, inclusion of this into the program logic is meant for future reference.

integer division is allowed and the range of variable values must also be between -32768 and 32767, the order and grouping of operations becomes very important. Thus, in line 35, the conversion from millimeters to gear teeth requires multiplying (mm) by 44 before dividing by 188. Otherwise, the result will always be zero.

Secondly, in lines 400-460, the conversion from degrees to actual counter values must be grouped in specific order of operations or extraneous errors will result. The equation to linearly map from a value X in domain [A,B] to another value Y in domain [C,D] is:

$$Y=(X-A)\,\frac{D-C}{B-A}+C$$

For example, for the steering servo, X might be 0° representing straight ahead. Domain [A,B] is then [-45°,45°]. Using the values of $0BB8 (3000) and $076C (1900) for D and C respectively[6], the equation would then make Y equal to $0992 (2450). The result is to be expected as Y is in the middle of the two extremes, D and C.

However, implementing this directly in 6811 BASIC would cause errant results. Thus, for example, dividing the term 'B-A' into sub-parts prevents the variables CV and DV from overflowing. This is what is done in lines 405 and 430 of the program.

[6] See Appendix C for the values of C and D.

In lines 55-80, the odometer counters are first cleared. Then, the data to spin the wheels in reverse is sent to Port C. However, the direction of travel is then checked to determine if reverse movement is actually what is desired. If forward movement is actually the case, line 70 is executed which resets Port C correctly. Note that the value used is $E6 rather than the maximum of $FF. This is because $E6 represents the same rotational speed as $00 does in reverse. Finally, the motor is engaged with a $01 to Port B.

As mentioned earlier, reading in the optical sensor counter (odometer) value requires modification of memory location $1002. Thus, lines 801-802 and 855-860 accomplish just that. Note that it is necessary to modify just bit 0 of $1002. Otherwise the DC motor and possibly even the servos will behave unexpectedly.

A listing of the variables used in the program is also included in Appendix D.

Discussion

This project was both interesting and frustrating because it was in the area in that I wanted to concentrate on. However, at various points during the semester, I felt no progress was being made. Furthermore, when I first started the project, I was under the impression that all the subsystems worked, especially the ultrasonic control board. I later realized that this was not true and in fact, the ultrasonic control board was what gave me the most trouble. The following is a list of the major problems I encountered.

At first, I was unsure as to whether the DC motor was strong enough to move the car. Upon further inspection, I found that the motor was adequate as long as the resistance from the various cables necessary (power, terminal interface) for operation was minimized. However, making the car totally self-contained with its own power source is not possible presently. I believe that for the car to be a separate unit, a stronger motor will have to be found.

Secondly, it was necessary to use the data books to figure out how to program the MC6840 chip as the timers were not functional at the beginning of the project. Programming the chip took a considerable amount of time since I had to do it by trial and error. Also, the servo for the transducer mount needed to be replaced since its internal wiring was malfunctioning. In replacing the servo, I realized that a potentiometer reading from the servos for analyzing their orientation was not necessary. This is because the servos have their own closed-loop feedback system for maintaining their direction. Thus, since the program will know precisely what angle the wheel and sensor mount are turned to because the program is the one that sent the commands, additional feedback will not provide any further information.

Since the other subsystems now function correctly, I will concentrate on discussing the ultrasonic control board and what I further understand of its operation. First, since the ultrasonic board was not working when I started the project, I decided to bypass the toggling between the two transducers as described in Vivek Shrivastava's report. I proceeded by wiring the VSW, XLG, XLG', and MFLOG lines directly to P2 thus avoiding the diagram on page 10 of his report. By skipping the added control logic, the question of whether the switching circuitry is to blame for the nonfunctioning ultrasonic sensors can be readily determined. I found, however, that doing the above procedure had no effect on the operation of the sensors. Note that the control board is currently still wired in this state and not as presented in Vivek's report.

I then examined the inputs and outputs to the Polaroid board to see if they were faulty. I found that the VSW signal was not switching high when a "start sonar signal" was initiated. The VSW's failure to change states was remedied by going through the logic schematic and tracing the circuit. By looking at the diagrams in the Polaroid manual, I also found that the wiring to the Polaroid boards were in error which contributed to the problem.

Since the board now seemed to be hooked up correctly, I attempted to examine whether or not the XLG signal, which should contain the 56 pulse cycles, was being generated correctly. An oscilloscope triggered by the VSW signal was used to investigate the problem. However, nothing substantial could be seen. As a result, I tried hooking up the Polaroid directly to a power source and not going through the control logic. From this, I was able to hear a "chirp" as described in the manual. However, I still could not see the 56 cycles on the oscilloscope. After trying three other Polaroid boards, I was still unable to get any signals to be generated from the boards. Therefore, I am at a loss as to what to try next.

Results

As stated in the project proposal, the goals of the project were to integrate some hardware control logic onto a 1/10 scale model car. Furthermore, the project required that all the subsystems function simultaneously. Finally, a control program was to be written which would handle the operations of the various devices given input parameters and would provide feedback as to the car's position in its environment.

The results that I obtained from work on the project achieved most of these goals. The car does move forward and backward, turn its steering wheel in the specified degree amounts, and turn the transducer mount. Feedback through the odometer is obtainable and is used in the software control program.

Table I - Distance w/out Applying Reverse Voltage

Forward	Forward	Reverse	Reverse
20 cm	40 cm	20 cm	40 cm
34	55	29	53
37	59	32	53
35	61	32	55
35	61	32	55
36	62	30	54
Average			
35	60	31	54
Average Error			
15	20	11	14

The first experiment that was performed was to move the car forwards and backwards for distances of 20 and 40 centimeters. The results are shown in Table I. The data seems to suggest that the car moves in a consistent amount each time although errors exist between what is desired and obtained. Furthermore, increasing the desired distance from 20 cm to 40 cm does not increase the average error by a significant amount which leads me to believe that the errors are due to the momentum of the car which keeps it moving a constant amount even after a stop command has been issued.

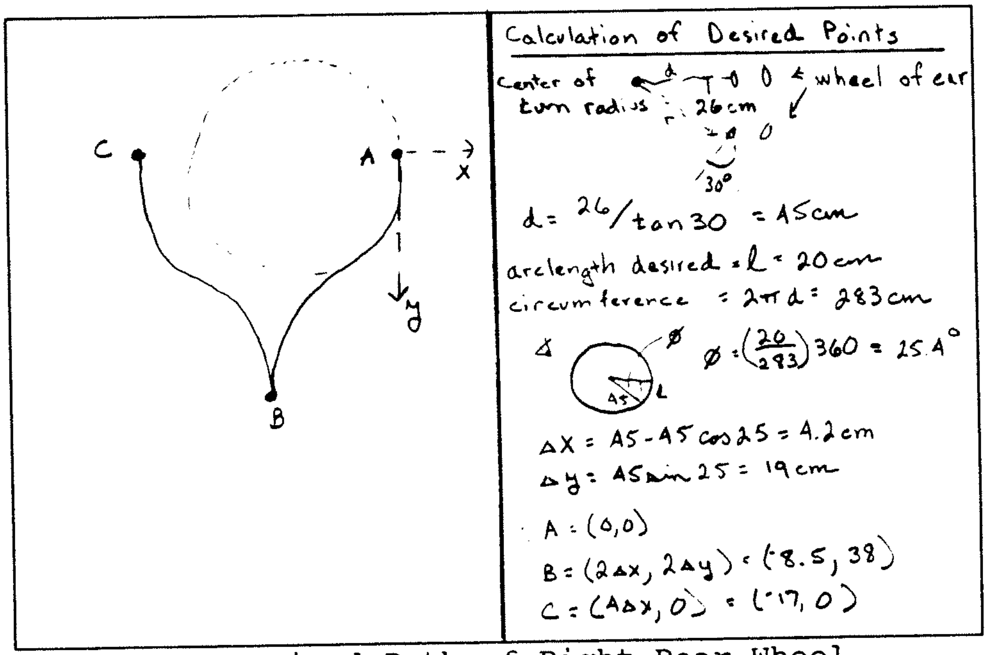
One way of making the car perform better is by applying a voltage to make the car move in the reverse direction as soon as the required distance has been travelled. Table II summarizes the data from doing just that.

Table II - Distance with Applying Reverse Voltage

Forward 20 cm	Forward 40 cm	Reverse 20 cm	Reverse 40 cm
27	53	27	51
28	52	28	52
28	51	26	51
26	53	27	53
27	52	27	51
Average			
27	52	27	52
Average Error			
7	12	7	12

The results from Table II are in much closer agreement to the desired values. Also note that the errors in forward and reverse movement are in better agreement than in Table I. I believe this occurs because the DC motor inherently has more friction spinning in the reverse direction than in the forward direction. As a result, forward movement will generally have a higher error value. By applying a voltage in the opposite direction of travel, the friction has now been equalized in both paths. The error that is left could be due to conversion error in line 35 of the program or even a miscount by the optical sensor.

The last experiment performed was to make the car move in an S-like pattern. Automated parallel parking would require such a maneuver. Figure 1 shows the desired path of the right rear wheel of the car.



Calculation of Desired Points

$d = 26/\tan 30 = 45\text{cm}$

arclength desired $= l = 20\text{cm}$

circumference $= 2\pi d = 283\text{cm}$

$\emptyset = \left(\frac{20}{283}\right)360 = 25.4^\circ$

$\Delta X = 45 - 45\cos 25 = 4.2\text{cm}$

$\Delta y = 45\sin 25 = 19\text{cm}$

$A = (0,0)$

$B = (2\Delta x, 2\Delta y) = (-8.5, 38)$

$C = (4\Delta x, 0) = (-17, 0)$

Figure 1 - Desired Path of Right Rear Wheel

The car would start from point A, travel to point B, and then go on to point C. Then, the car would repeat the path but in the opposite direction to get back to the starting point. The program listing in Appendix D performs the necessary series of instructions. Using some trigonometry, point B is roughly at location (-8.5, 38)[7] while point C is at location (-17, 0). However, experimental values resulted in values of (-18, 29) and (-30, 4) for B and C respectively. In repeating the path, the car went to location (3, -1) instead of (0, 0) as desired.

One source of error could be that a rear wheel was used to obtain measurements rather than a front one. Since the rear wheel doesn't actually turn at an angle of 30° relative to the axis of the car, the value used in calculating the desired location value would be in error. Furthermore, these results are hard to interpret as the odometer is actually on the left rear wheel which slips depending on the direction the car is turning.

[7] All values are in centimeters.

Conclusion

In conclusion, I am pleased with the results that I obtained. Obviously, it would have been much better if the ultrasonic control board had worked. Evidently, then, more work is required before the car can be used as a test bed for the automated parallel parking algorithm.

Another item that needs to be addressed is the location of the odometer. According to Doug Lyon, the Ph.D. candidate developing the above mentioned algorithm, the optical sensor needs to be moved to one of front wheels since the path travelled by the front of the car is considerably different than that of the rear axle. Thus, the algorithm would not function correctly.

Furthermore, I have found that the braking method used to stop the car requires additional work. At present, the voltage applied to the DC motor is simply set to 0V when a stop command is issued. However, this method will not always yield correct results since the momentum of the car can force it to travel a greater distance than specified. As a result, overshoot will occur, which then will cause considerable errors. One way of fixing this is applying a reverse voltage as I have explained. Another would be to switch the motor to a resistor when braking is desired. These are just a few of the design options which are available for other students to pursue.

References

Ciaria, S. "Home on the Range!", Byte Magazine, November 1980. Byte Publications Inc.

Kordon, Mark, 6811 Computer Car Final Report, Spring 1991.

Motorola, M68HC11EVB Evaluation Board User's Manual, Motorola Distribution; P.O. Box 20912, Phoenix, AZ 85036.

Motorola, BASIC11 Reference Manual.

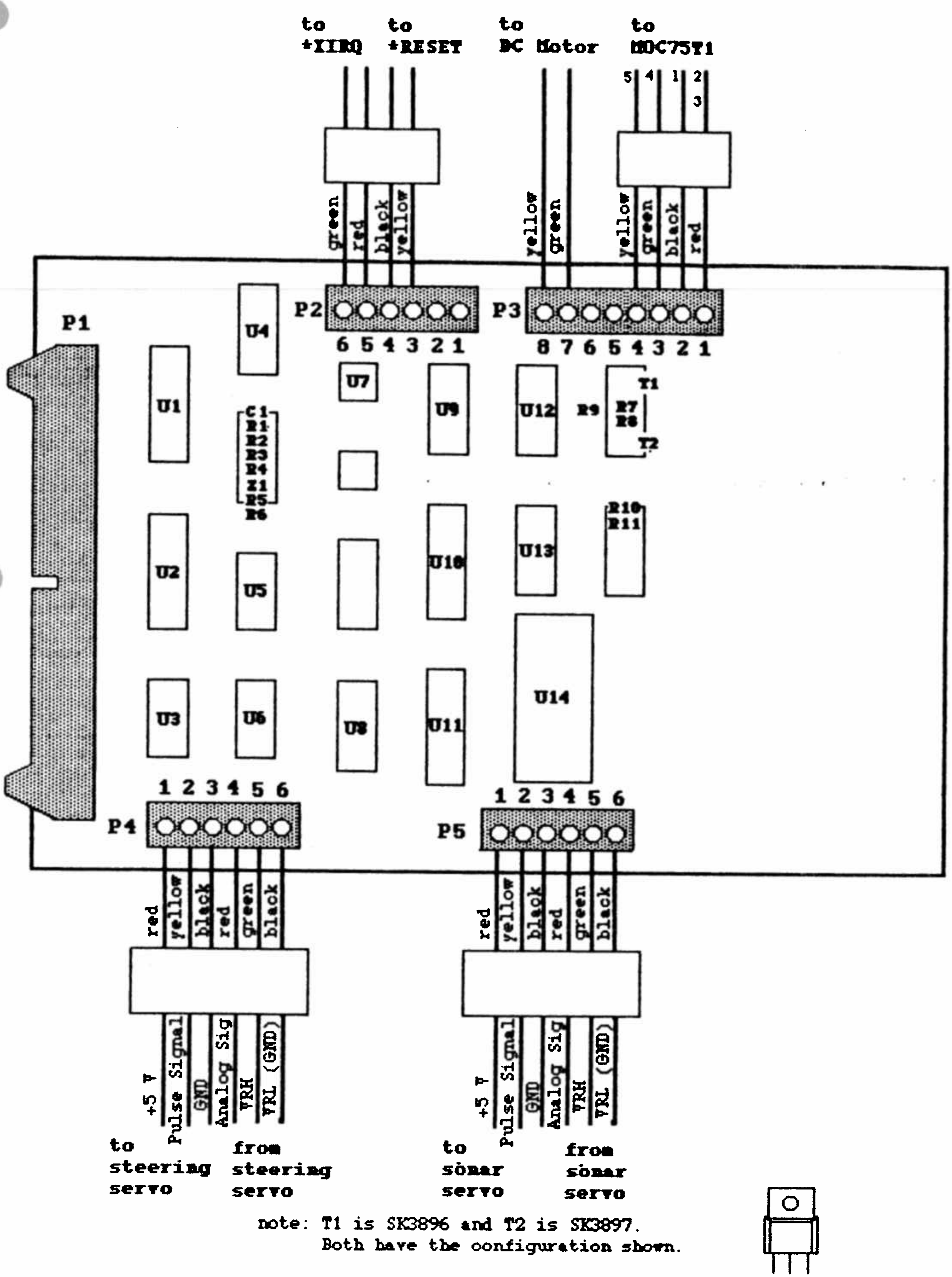
Polaroid, Ultrasonic Ranging System, Polaroid Corporation, Commercial/Battery Division, 575 Technology Square - 3, Cambridge, MA 02139.

Shrivastava, Vivek, Final Report on the Car Sensor Circuitry and Orientation, Summer 1991.

Appendix A - Motor Control Board Schematic

Note: These pages were photocopied from Mark Kordon's report pp. 16-19. Some values were changed to match the actual design.

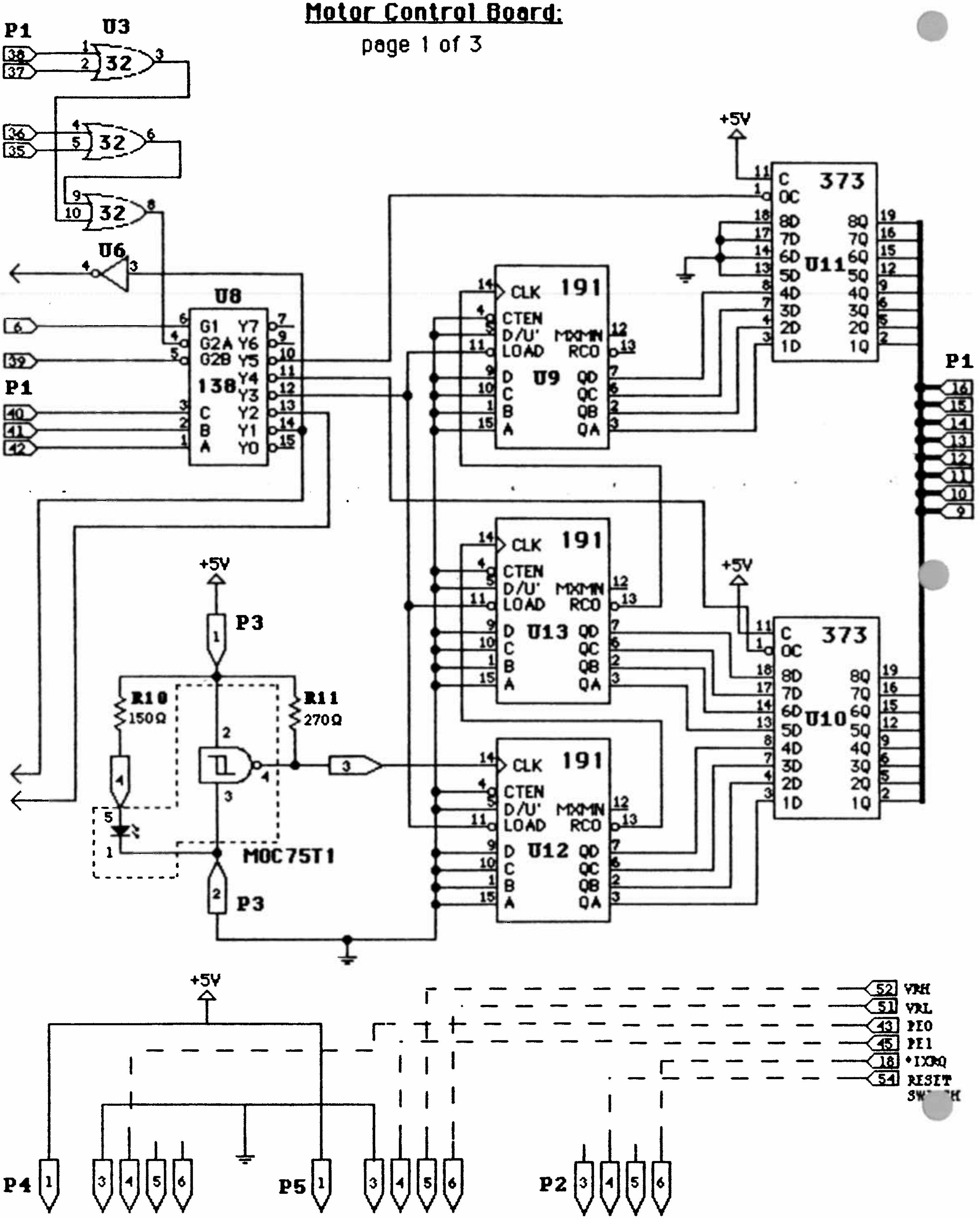
Motor Control Layout:



note: T1 is SK3896 and T2 is SK3897.
Both have the configuration shown.

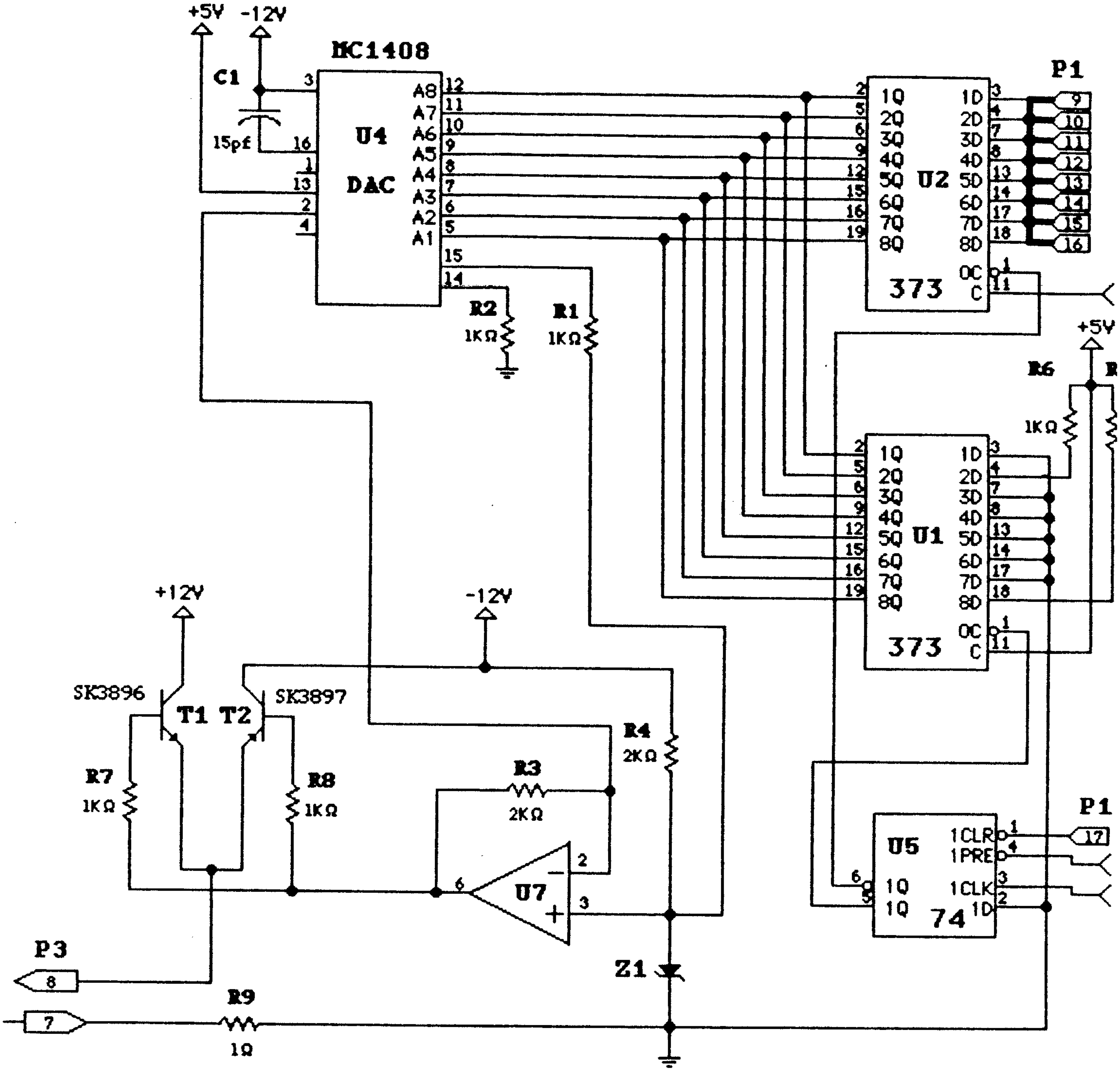
Motor Control Board:

page 1 of 3



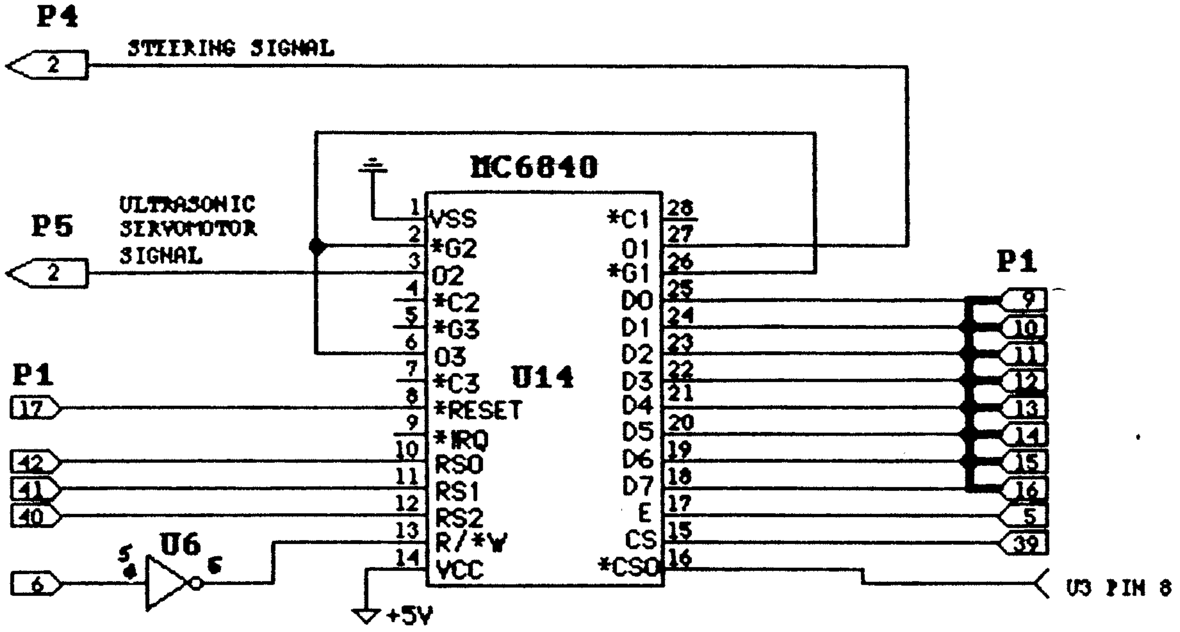
Motor Control Board:

page 2 of 3



Motor Control Board:

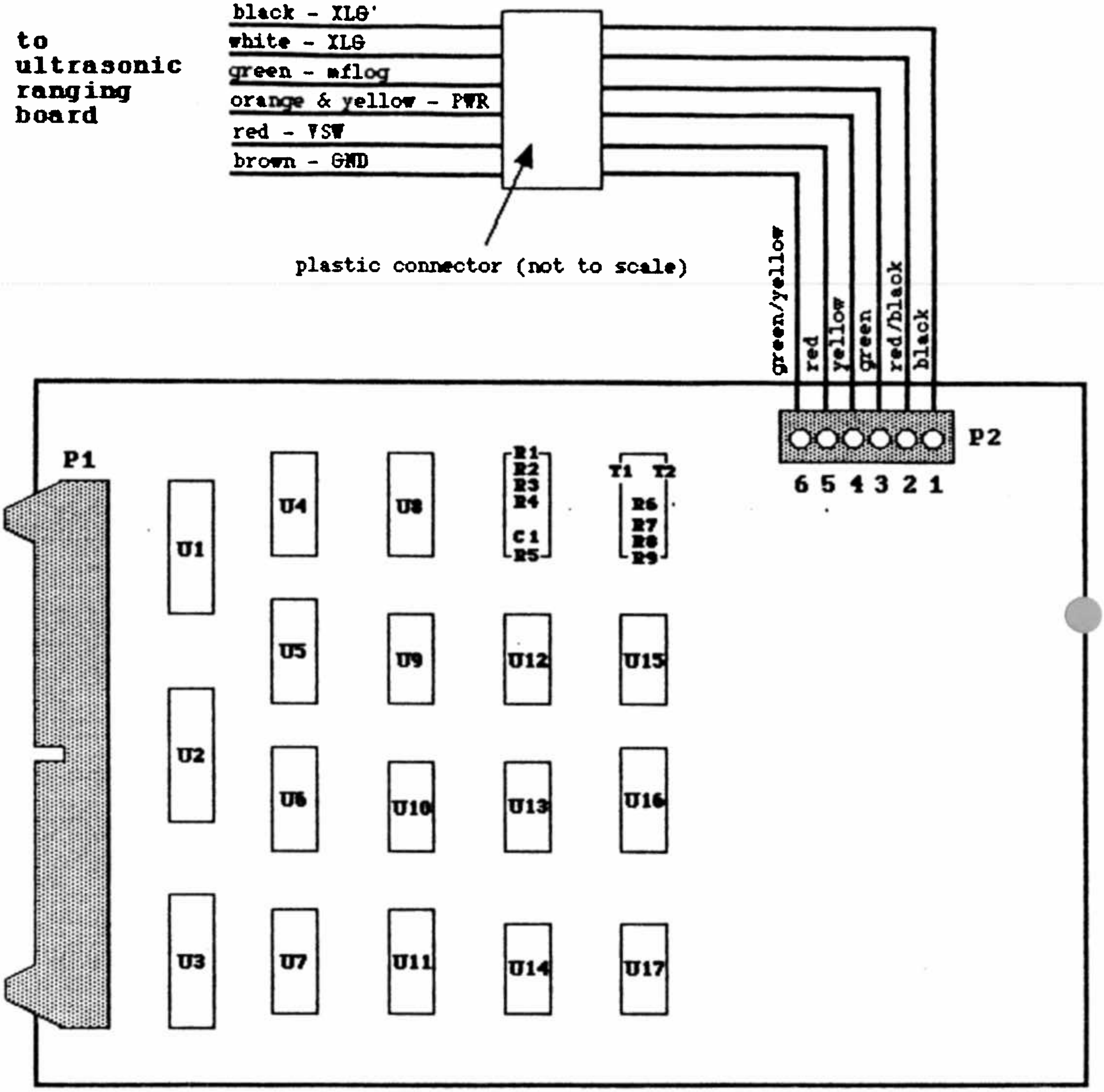
page 3 of 3



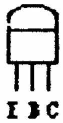
Appendix B - Ultrasonic Control Board Schematic

Note: These pages were photocopied from Mark Kordon's report pp. 23-25. Some values were changed to match the actual design.

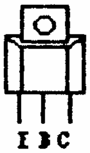
Ultrasonic Board Layout:



note: T1 is a 2N4401

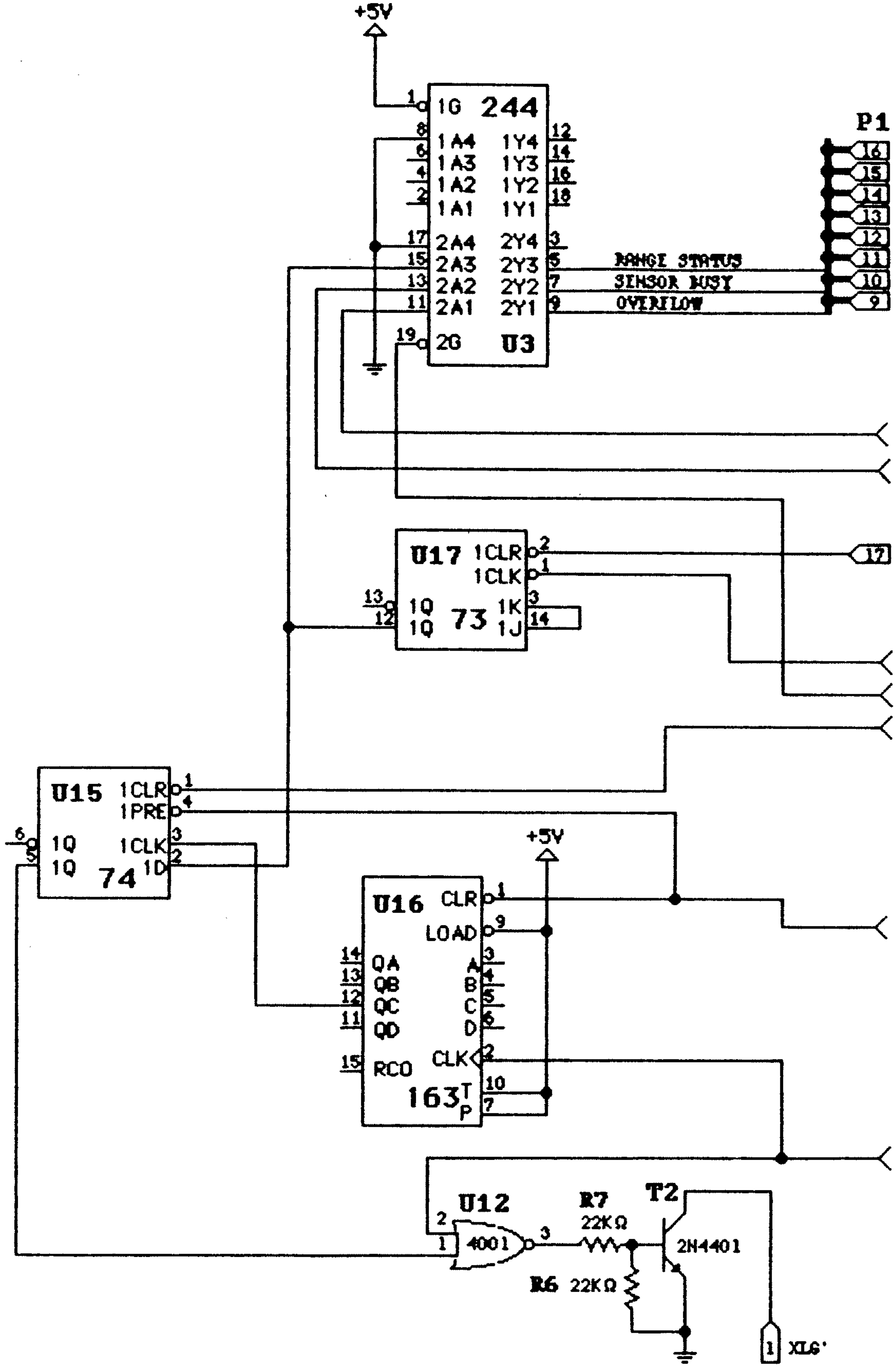


T2 is a MPU295



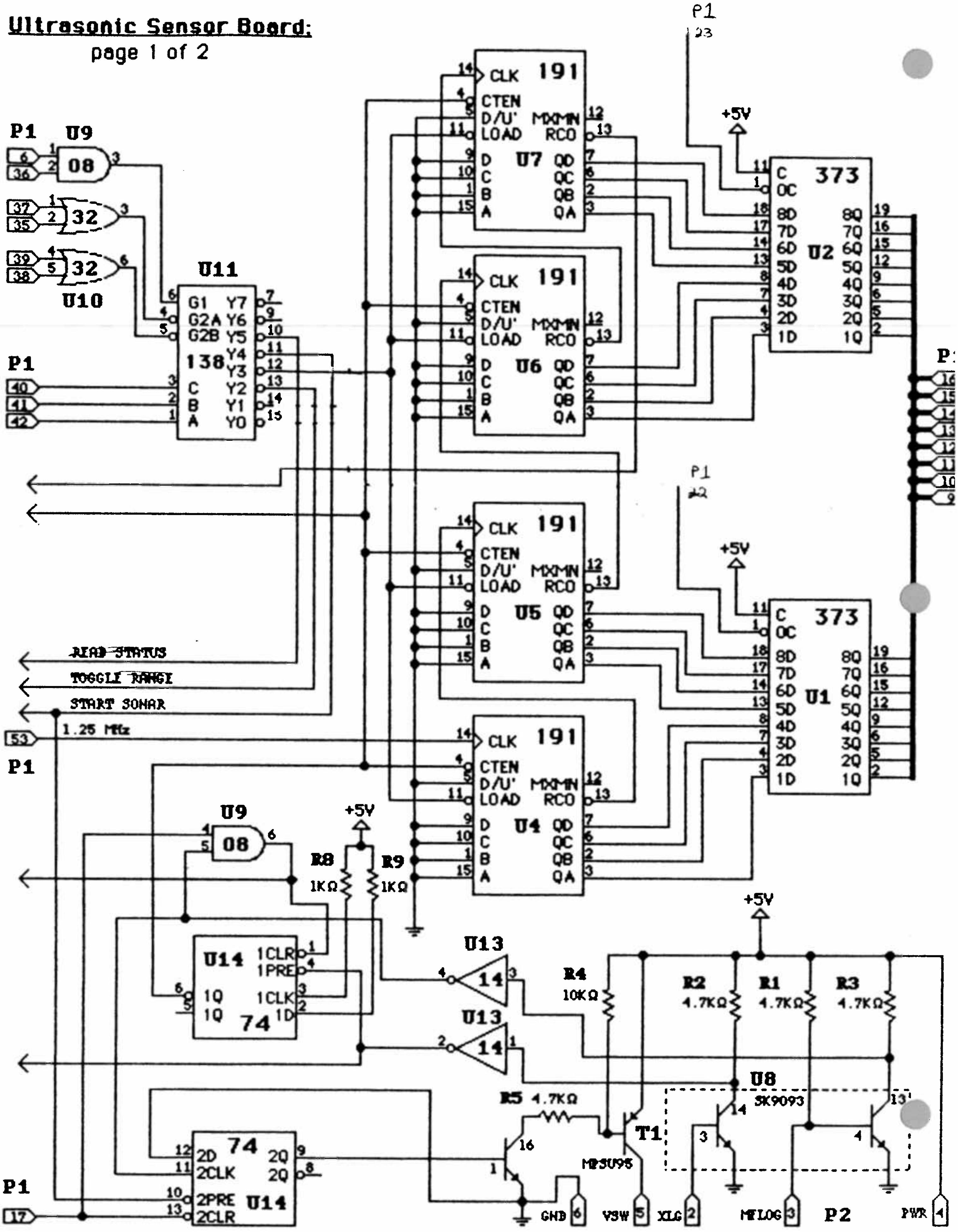
Ultrasonic Sensor Board:

page 2 of 2



Ultrasonic Sensor Board:

page 1 of 2



Appendix C - Pseudocode for Servos

Location	Value	Comments
$1007	$FF	set DDRC to output
$1003	$00	initialize port C
$1004	$09	allow write to control reg. 3
$1003	$4E	set up square wave of 20 msec
$1004	$0E	write to MSB latch of timer 3
$1003	$20	continue data for 20 msec wave
$1004	$0F	write to LSB latch of timer 3
$1003	$82	set timer 3 to continuous mode
$1004	$08	write to CR3
$1003	$0A*	MSB data for timer 2
$1004	$0A	write to MSB latch of timer 2
$1003	$F8*	LSB data for timer 2
$1004	$0D	write to LSB latch of timer 2
$1003	$A3	set timer 2 to single shot mode
$1004	$09	write to CR2
$1003	$09*	MSB data for timer 1
$1004	$0A	write to MSB latch of timer 1
$1003	$92*	LSB data for timer 1
$1004	$0B	write to LSB latch of timer 1
$1003	$A2	set timer 1 to single shot mode
$1004	$08	write to CR1 and let timers run

Data values for control of the steering and sonar values are:

	Left	Middle	Right
Steering	$0B B8	$09 92	$07 6C
Sonar	$10 F0	$0A F8	$05 00

The values between the left and right positions seem to follow a linear relationship which suggests that reading the potentiometer in the servo to get the servo direction would be unnecessary. By varying the values marked with an * above, the direction of the servo can be changed. Note that timer 2 is for the transducer mount servo and timer 1 is for the steering servo.

Appendix D - Listing of Software Program

This is a listing of the variables that are used in the program listed on the next page. It was necessary to use such short variable names since BASIC11 can only represent variables by two characters.

Variable	Description
B	Distance to travel in millimeters
C	Degrees to turn steering wheels
D	Degrees to turn transducer mount
BV	Distance to travel in gear teeth (absolute)
E	Distance travelled so far in gear teeth (absolute)
CV	Counter value to set duty cycle for steering servo
CH	High byte of CV
CL	Low byte of CV
DV	Counter value to set duty cycle for trans. servo
DH	High byte of DV
DL	Low byte of DV
Y	Temporary value of memory location $1002

```
NEW
10 READ B
20 IF (B=0) THEN 900
30 READ C,D
35 BV=ABS(B)*44/180
40 GOSUB 400
50 GOSUB 500
55 POKE ($1004,$03)
60 POKE ($1003,$00)
65 IF (B<0) THEN 80
70 POKE ($1003,$E6)
80 POKE ($1004,$01)
90 GOSUB 800
110 IF E<BV THEN 90
115 PRINT CHR$(7)
120 GOTO 10
400 REM CONVERT FROM DEGREES TO ACTUAL DATA NUMBERS
401 C=-C
405 CV=(((C+30)*((3000-1900)/5))/12)+1900
410 CH=CV/256
420 CL=CV-CH*256
425 D=-D
430 DV=(((D+90)*((4336-1280)/30))/6)+1280
440 DH=DV/256
450 DL=DV-DH*256
460 RETURN
500 REM TURN STEERING WHEEL AND ULTRASOUND SENSOR SERVOS
501 POKE ($1007,$FF)
510 POKE ($1003,$00)
520 POKE ($1004,$09)
530 POKE ($1003,$4E)
540 POKE ($1004,$0E)
550 POKE ($1003,$20)
560 POKE ($1004,$0F)
570 POKE ($1003,$82)
580 POKE ($1004,$08)
590 POKE ($1003,DH)
600 POKE ($1004,$0H)
610 POKE ($1003,DL)
620 POKE ($1004,$0D)
630 POKE ($1003,$A3)
640 POKE ($1004,$09)
650 POKE ($1003,CH)
660 POKE ($1004,$0A)
670 POKE ($1003,CL)
680 POKE ($1004,$0B)
690 POKE ($1003,$A2)
700 POKE ($1004,$06)
710 RETURN
800 REM READ IN DISTANCE TRAVELLED IN GEAR COUNT
801 Y=PEEK($1002)
802 POKE ($1002,(Y.AND.1))
805 POKE ($1007,$00)
810 POKE ($1004,$05)
820 EH=PEEK($1003)
830 POKE ($1004,$04)
840 EL=PEEK($1003)
850 E=EH*256+EL
855 Y=PEEK($1002)
860 POKE ($1002,(Y.OR.1))
870 POKE ($1007,$FF)
880 RETURN
900 REM END OF PROGRAM
901 POKE ($1004,$02)
910 END
1000 REM DATA TO BE READ IN
```

```
1001 DATA 250,30,0
1010 DATA 250,-30,0
1020 DATA -250,30,0
1030 DATA -250,-30,0
1040 DATA 250,-30,0
1050 DATA 250,30,0
1060 DATA -250,-30,0
1070 DATA -250,30,0
1080 DATA 0
```

Appendix E - Helpful Hints and Suggestions

Listed below are some helpful hints and suggestions which might help the next student in continuing this design project.

Hardware

The main connections from the EVB to the motor control board should not pose any problem. However, make sure that the red and red-black terminals of the DC motor go to the yellow and green terminals of the control board respectively. This will enable the DC motor to turn in the proper direction.

The optical sensor wires seem to be too short for what is needed. Sometimes, they could come off the sensor completely. As a result, no feedback is sent to the EVB and the program will have no way of sensing how far the car has gone.

Currently, both Ports B and D are required to control the ultrasonic control board. However, a conflict on the data line (Port C) might occur if data is being sent out on the line by either the EVB or motor control board while the outputs of modules U1 and U2 of the ultrasonic control board are enabled. To remedy this situation, make sure that U1 and U2 are disabled if their outputs are not required. A better way to do this is to rewire the circuit as shown in Mark Kordon's report thus automatically disabling U1 and U2 when communication to the motor control board is desired.

The user can talk to the EVB through either a dumb terminal or through the Macintosh using the Milford cable. If using a dumb terminal, connect one end of an RS-232 cable to the P2 port of the EVB and the other to the EIA port of the terminal. If using the Milford cable, connect one end to the Macintosh's communications port and the other to P2 of the EVB. An extension cable is supplied with the project. This cable converts the 25 pin connector to the 15 pin connector mounted on the car. Using this cable allows the car to move a greater distance.

Software

Communicating to the car is done using a terminal program called Versaterm-Pro V3.0.1. This program is copyrighted and should be used exclusively for this project only. A disk with this program and the BASIC listing is included with this final report. The user simply needs to double-click on the Versaterm icon after inserting the disk.

Sending the program to the EVB is done through the "Send Stream" option under the File command. Editing of the program can be done through the "Edit Text" option under the "Edit" command. More extensive help can be found through the online help that is available in the program.