Introduction

The objective of the Robotic Navigation Distance Control Platform is to design and build a robotic platform that maintains a fixed safety distance behind another moving object to avoid collisions. The robotic platform contains an EMAC 80515 microcontroller that interfaces a distance sensor, electric motor and steering on the robotic platform.

Applications

This can be useful in such applications as robotics, in which there might be several robots that are required to follow one another, and this system would prevent them from colliding. Another application is in the automotive industry, which a similar device can be installed on a vehicle to maintain a safe distance behind another vehicle. This would be useful when a vehicle using its cruise control approaches a slower moving vehicle. When the faster moving cruise control vehicle approaches the slower moving vehicle, this safety distance system would slow the faster moving vehicle to match the speed of the slower moving vehicle to prevent an accident (see Figure 1.1). This vehicle adaptive cruise control system is used in such vehicles as Lexus and Mercedes-Benz. This technology is used to prevent collisions while the driver is using the cruise control on the vehicle.

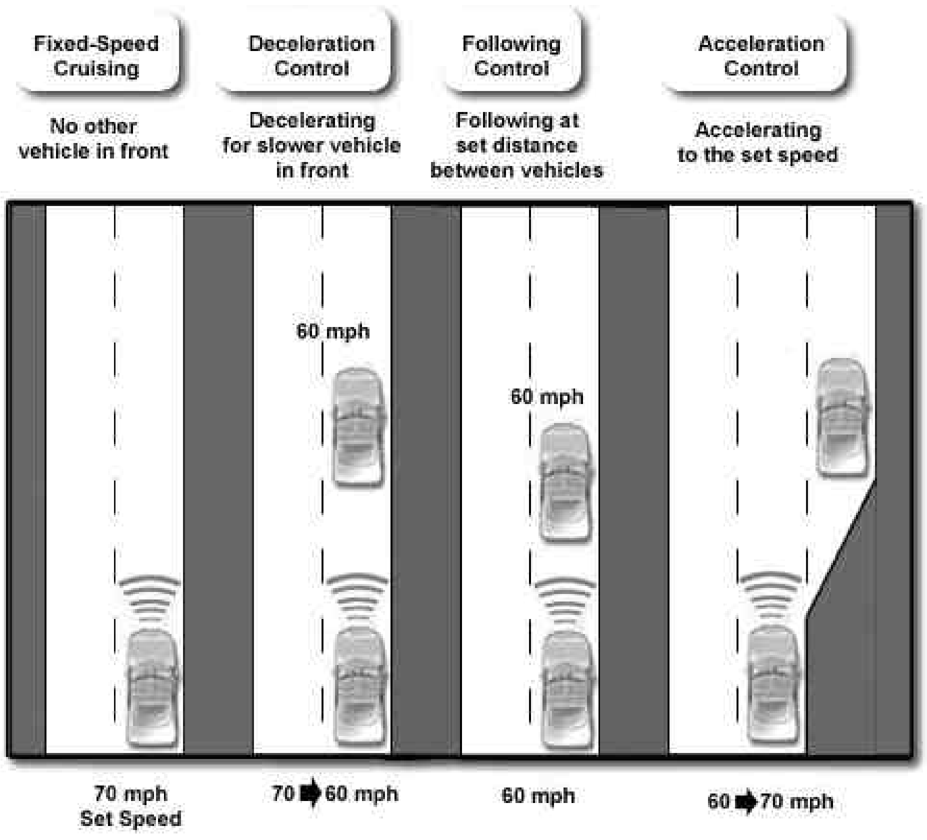

Fixed-Speed Cruising
No other vehicle in front
Deceleration Control
Decelerating for slower vehicle in front
Following Control
Following at set distance between vehicles
Acceleration Control
Accelerating to the set speed
60 mph
60 mph
70 mph Set Speed
70 ➡ 60 mph
60 mph
60 ➡ 70 mph

Figure 1.1
Adaptive Cruise Control on Vehicles
Source: http://www.ece.msstate.edu/classes/design/ece4532/2003_spring/cruise_control/

The Robotic Navigation Distance Control Platform keeps a constant distance from a moving object through software and hardware development. This Robotic Platform has six different software modes of operation: Fixed Navigation Mode, User Out of Range Mode, Auto Out of Range Mode, Stop / Reload Mode and Navigation Control Mode, Increment / Decrement Mode. The hardware aspect of this project consists of inputs and outputs to the robotic platform and hardware subsystems. The modes of operation and hardware will be discussed below.

Modes of Operation

Fixed Navigation Mode:
All systems are powered and the robotic platform waits for the user to enter a fixed safety distance in feet to follow the object. First, the user is asked to enter either User or Auto Out of Range Mode. Second, the user enters the desired distance, and then presses the activation button on the keypad to activate the robotic platform navigation controls. The robotic platform then proceeds to navigate behind a moving object.

User Out of Range Mode:
If the object being followed is out of range or there is no signal from the sensor, the robotic platform enters an Out of Range Mode in which the robotic platform stops. The EMAC microcontroller displays “Out of Range”. The robotic platform then waits for the user to reactivate the navigation controls, which then activates the navigation controls and displays “Following” on the LCD screen.

Auto Out of Range Mode:
This mode is similar to User Out of Range Mode except the robotic platform activates the navigation mode once an object is placed back within range of the sensors. The EMAC microcontroller displays “Following” on the LCD screen.

Stop / Reload Mode:
User is able to stop and reload the motor speed manually using keypad input.

Navigation Control Mode:
User is able to activate or deactivate the Fixed Navigation Mode.

Increment / Decrement Mode:
User is able to increment or decrement motor speed by one unit manually.

System I/O

There are several inputs and outputs to the EMAC Microcontroller. The inputs and outputs are described below. See Table 3.1 and Figure 3.1 for complete inputs and outputs to the EMAC microcontroller.

Inputs	Outputs
User Keypad	LCD Display
Distance Control Sensor	Robotic Platform Motor
	Robotic Platform Steering

Table 3.1
System Inputs and Outputs

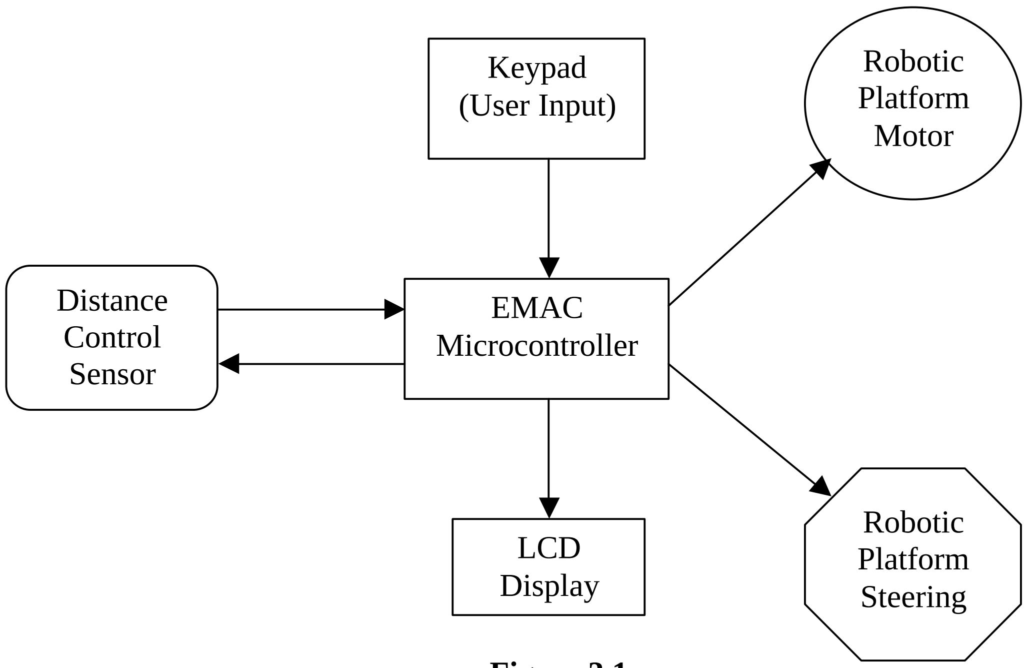


Figure 3.1
System I/O Block Diagram

Inputs to EMAC Microcontroller:

User Keypad:
When in navigation mode, the user enters the required fixed following distance and the Out of Range Mode and activates the robotic platform using the keypad. If the robotic platform enters the User Out of Range Mode, the user must press activate button to reactivate robotic platform navigation mode. The keypad also allows the user to manually stop, reload, increment and decrement the electric motor speed.

Distance Control Sensor:
The distance sensor consist of an ultrasonic SRF04 sensor that will be mounted on the robotic platform. The distance sensor will point straightforward and parallel to the ground, which will determine distance behind the moving object. The EMAC will control the electric motor on the robotic platform from the signal received from the ultrasonic sensor.

Outputs from EMAC Microcontroller:

LCD Display:
The LCD display shows the mode of operation that the robotic platform is currently performing. The LCD display will also provide information for the user, allowing the user to enter the different manual modes of operation. Finally the LCD will be used as a display prompt to ask the user to enter in the desired safety distance in feet as well as to enter the Out of Range Mode. See Figure 4.1 for location of distance sensor on robotic platform.

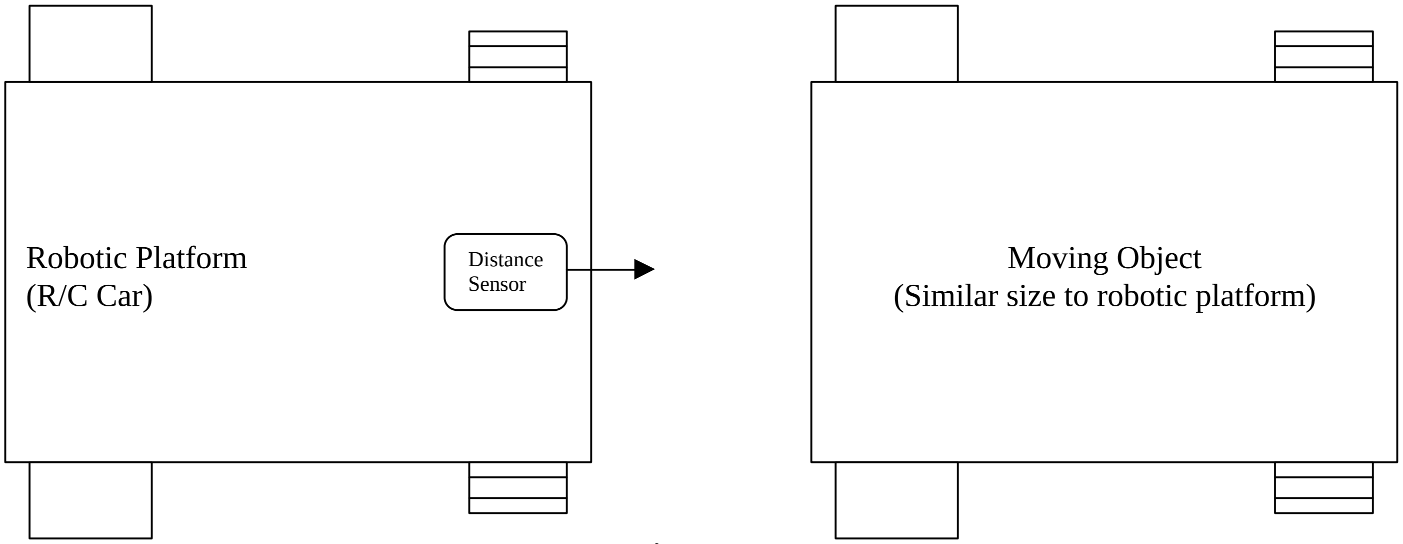


Figure 4.1
Distance Control Sensor Diagram

Robotic Platform Motor:
The EMAC microcontroller controls the motor speed, allowing the robotic platform to safely follow the moving object.

Robotic Platform Steering:
The EMAC microcontroller controls the steering of the robotic platform to navigate behind the moving object. **Note:** The steering is not implemented, and is kept in the neutral position.

Hardware

The hardware subsystems consist of a distance control sensor subsystem, an electric motor subsystem and a steering system. The robotic platform chosen for this project is a radio controlled R/C car. The sensors will be mounted on the R/C car and the current R/C car's electric motor and servomotor will be used for navigation control. Each subsystem is discussed in full detail below. See Figure 5.1 for hardware subsystem block diagram.

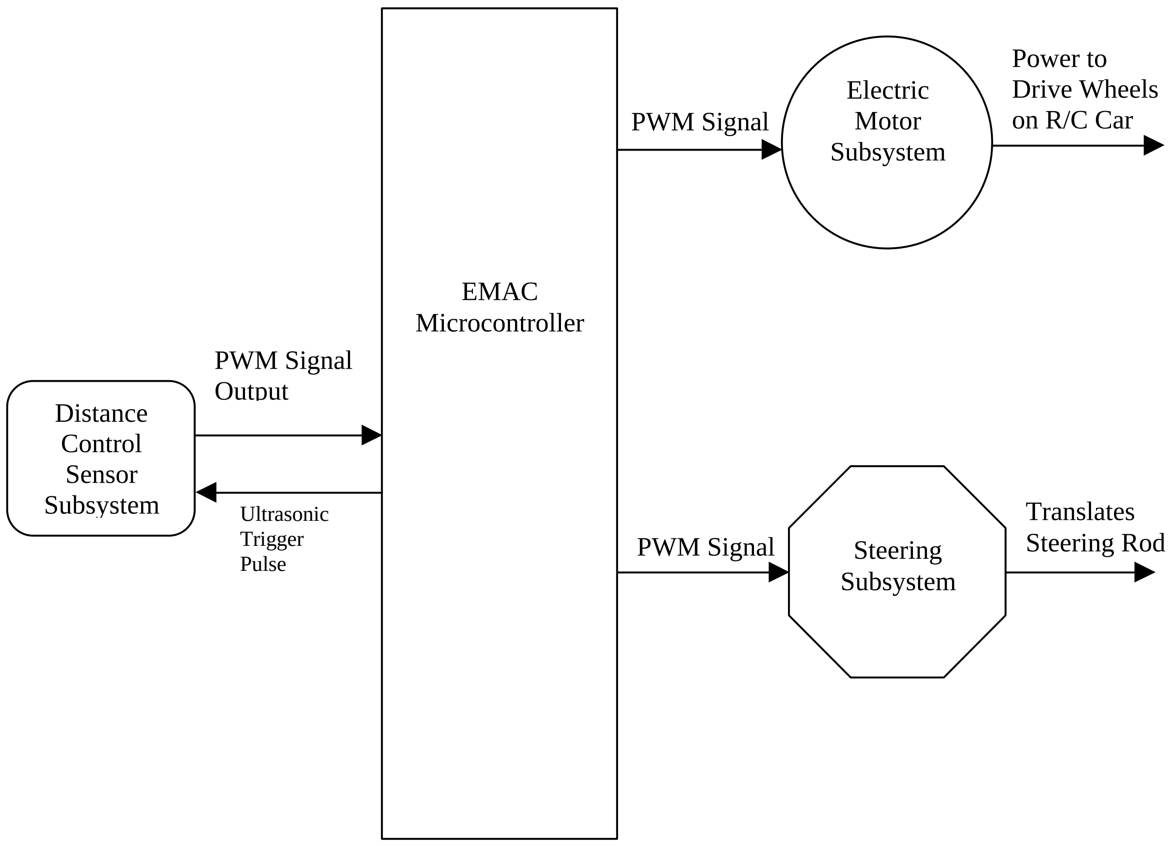


Figure 5.1
Hardware Subsystem Block Diagram

Distance Control Sensor Subsystem

Input Signal to Ultrasonic Sensor:
A trigger pulse of 1.5 ms from the microcontroller is used to start the initial sequence, which transmits an ultrasonic pulse. If an object is in front of the sensor, the transmitted wave reflects off the object, and the same sensor will receive the reflected wave. See Appendix C1 for SRF04 timing diagram and complete data sheet.

Sensor Output Signal:
The output signal from the sensor is related to the distance between the sensor and the object in front of the sensor. The output from the sensor is a pulse width modulation (PWM) signal with a large pulse width related to a large distance.

Electric Motor Subsystem

Input Signal to Motor:
The input signal to run the motor consists of a PWM signal provided from the microprocessor, which controls revolutions per minute (RPM) of the motor. The PWM signal consists of a 33Hz signal with the positive pulse width varying from 1.0ms to 1.7ms. The 1.0ms pulse width is the neutral position for the electric motor and the 1.7ms pulse is full speed of the electric motor. See Appendix D for specifications, definitions and operating instructions for the Team Novak Rooster ESC. **Note:** The electric motor reverse is not implemented.

Motor Speed Output:
The motor shaft drives a gearbox that connects to the wheels of the robotic platform. Depending on the input pulse width of the PWM signal, the motor's shaft speed varies, providing the different ground speeds for the robotic platform.

Steering Subsystem

Input Signal:
The input signal consists of a PWM signal from the microcontroller and the variations in the input PWM signal will control a servomotor. The PWM signal consists of a 33Hz signal with the positive pulse width varying from 1.1ms to 1.9ms with 1.5ms as the servo's centering position. See Appendix E for Hitec HS-303 servo specifications. **Note:** The steering is not implemented, and is kept in the neutral position.

Output Steering Rod:
The steering rod connects to the servo horn, which is a plastic lever arm attached to the servomotor. The rotational movement of the servo horn produces a translation movement. The other end of the steering rod connects to the wheel linkage that controls the robotic platform's direction.

Hardware Circuit Diagram

The circuit diagram that connects the SRF04 ultrasonic sensor, the electronic speed controller and the servo to the 80515 microcontroller is seen in Figure7.1. The 5 Vdc for the ESC connector, servo connector and the SRF04 were connected to a voltage regulator. Even though the microcontroller board would be able to power these three devices, a voltage regulator was added to power these devices to protect the microcontroller board and to allow for future expansion to the project.

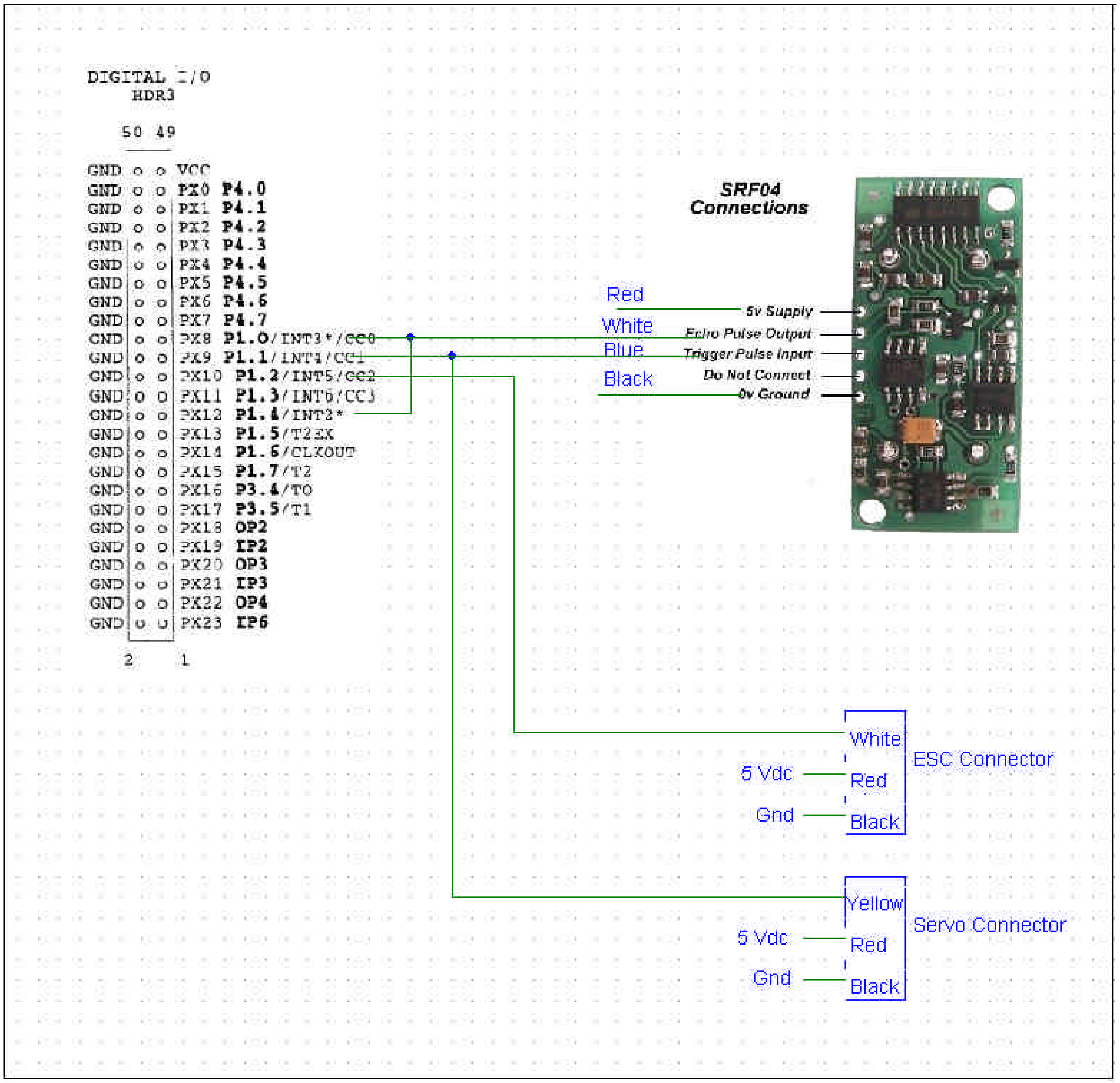


Figure 7.1
Hardware Circuit Diagram

Software

The software programmed on the EMAC 80515 microcontroller processes the incoming distance signal. The EMAC microcontroller software provides appropriate PWM signal to the electric motor subsystem to allow the robotic platform to maintain the desired safety distance from the moving object. The different modes of operation are: Fixed Navigation Mode, User Out of Range Mode, Auto Out of Range Mode, Navigation Control Mode, Increment / Decrement Mode and Stop / Reload Mode. These modes of operation are discussed in more detail with software flow charts below.

Fixed Navigation Mode:

All systems are powered, and the robotic platform asks the user to enter the desired safety following distance from 1 to 9 feet. The EMAC waits for the user to enter a fixed safety distance in feet through the keypad. The user will then enter the User or Auto Out of Range Mode. Last the user will press the activation button on the keypad to activate the robotic platform navigation controls. The robotic platform will then proceed to navigate behind a moving object.

The Fixed Navigation Mode software flowchart is followed and is shown in Figure 18.1 and 19.1 in Appendix A. If the control bit is 1 after Check Control Variable in Figure 19.1, the right flowchart is followed. If the object is out of range then the user selected Out of Range Mode is called. If the object is in range of the robotic platform then, the fixed distance control function determines if the motor speed should increase, decrease or kept constant. If the control bit is 0, then the right software flowchart is bypassed, allowing the fixed navigation mode to be deactivated. If a manual mode key is pressed on the keypad as seen in Figure 19.1 the appropriate software mode is called. When the navigation control button (0) is pressed the Navigation Control Mode is called, which the software returns back to the Check Control Variable in Figure 19.1.

The Fixed Navigation Mode uses the Fixed Distance Control function in Figure 19.1 to keep the distance between the robotic platform and the followed object constant. This is accomplished by calculating the distance between the robotic platform and the followed object from the distance sensor signal. The calculated distance will be compared to the user specified distance and the motor subsystem will adjust to achieve the user specified distance. If the distance sensor does not detect an object or no signal is received the Out of Range Mode will be entered.

User / Auto Out of Range Modes:

If the object being followed is out of range or there is no signal from sensor, the robotic platform will enter an Out of Range Mode, in which the robotic platform will stop. The EMAC microcontroller will display "Out of Range" on the LCD. The User Out of Range Mode will wait for the user to reactivate the navigation controls, which "Following" is then displayed on the LCD screen.

The Auto Out of Range Mode is similar to user Out of Range Mode except the robotic platform will continue navigation once an object is placed back within range of the sensors. The EMAC microcontroller will then display "Following" on LCD screen.

If the User or Auto Out of Range Mode is called from the Fixed Navigation Mode software seen in Figure 19.1 in Appendix A, the User / Auto Out of Range Mode software Flowchart in Figure 9.1 below is followed. In the flowchart the Stop Electric Motor function will stop the electric motor to prevent the platform from crashing into an object. If the user selected the User Out of Range Mode, the left software flowchart is followed, after "Out of Range" is displayed on the LCD screen seen in Figure 9.1. In the User Out of Range Mode the platform waits for the user to press the button 0, which calls the Navigation Control Mode. If the user selected the Auto Out of Range Mode then the right software flowchart is followed after the "Out of Range" is displayed on the LCD screen seen in Figure 9.1 In the Auto Out of Range Mode, the robotic platform waits until the distance sensor detects an object. Once an object is detected, the LCD displays "Following" and the software returns back to the Check Control Variable in the Fixed Navigation Mode seen in Figure 19.1.

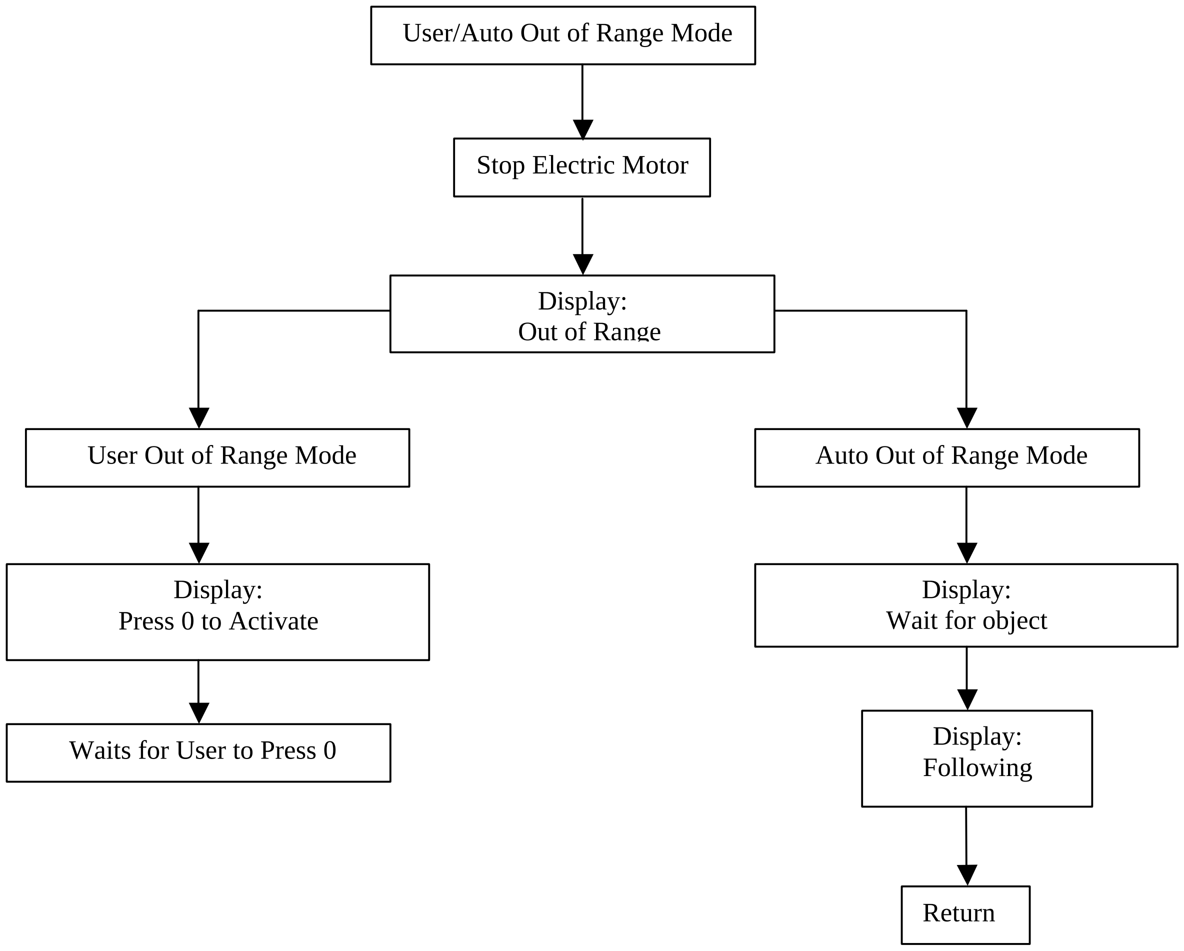


Figure 9.1
User / Auto Out of Range Mode Software Flowchart

Stop / Reload Mode:
User is able to stop and reload the motor speed manually using keypad input. If the user presses the stop button (B) on the keypad, the Stop software is called and follows the software flowchart on the left in Figure 10.1. If the Reload Motor Speed Button (D) is pressed on the keypad, the Reload Motor Speed software is called and the flowchart is on the right in Figure 10.1. At the end of both flowcharts there is a return, which means that both software modes return to the Check Control Variable in the Fixed Navigation Mode seen in Figure 19.1 in Appendix A.

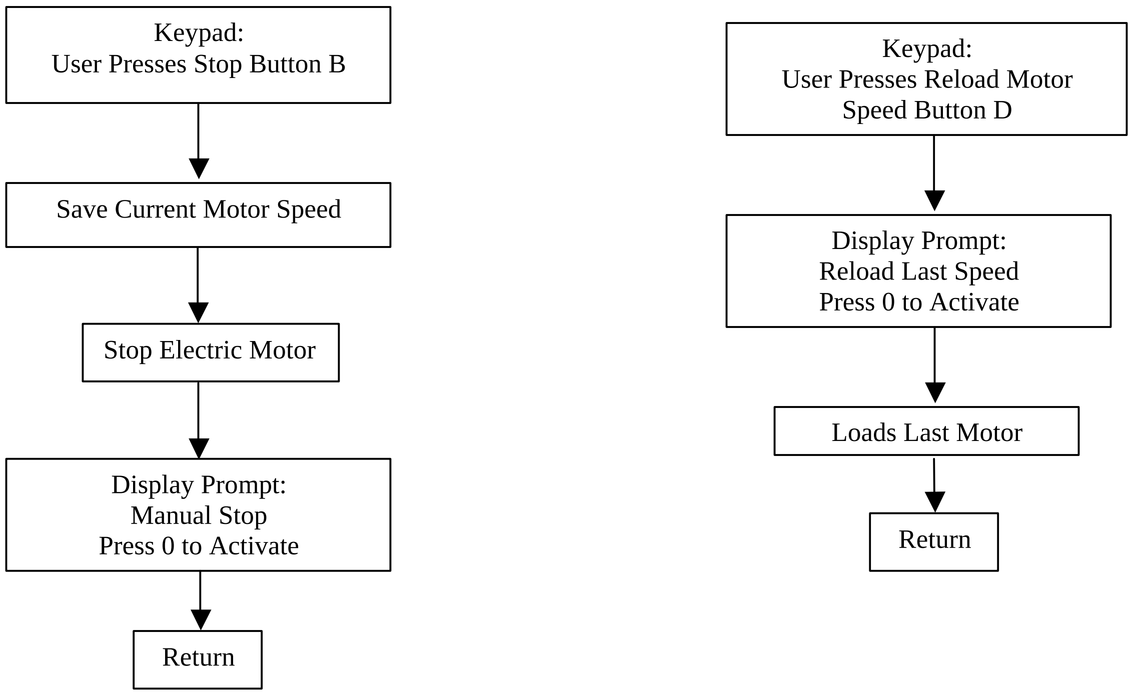


Figure 10.1
Stop / Reload Mode Software Flowchart

Increment / Decrement Mode:
User is able to increment or decrement motor speed by one unit manually. If the user presses the Increment Motor Speed Button (C) the software flowchart on the left in Figure 11.1 is followed. The IncMotorSpeed () function is the same function that the Fixed Navigation Mode uses to increase the Motor speed by one unit. When this function is called it increases the PWM signal to the Electric Motor Subsystem, which increases the motor speed. If the user presses the Decrement Motor Speed Button (E), the software flowchart on the right in Figure11.1 is followed. The DecMotorSpeed () function is the same function that the Fixed Navigation Mode uses to decrease the Motor speed by one unit. When this function is called it decreases the PWM signal to the electric motor subsystem which decreases the motor speed. At the end of both flowcharts there is a return, which means that both software modes return to the Check Control Variable in the Fixed Navigation Mode seen in Figure 19.1 in Appendix A.

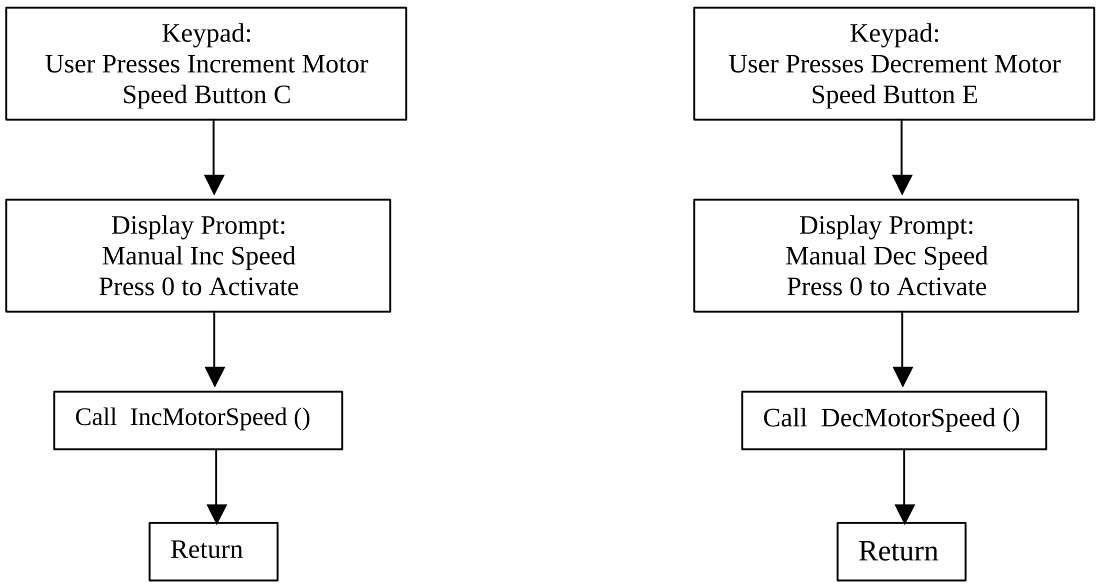


Figure 11.1
Increment / Decrement Mode Software Flowcharts

Navigation Control Mode:
User is able to activate or deactivate the Fixed Navigation Mode. When the user presses the Control Button (0) either in the Fixed Navigation Mode or in any of the manual modes this software flowchart is used. There is a variable control bit that is toggled when this software mode is called. When the Robotic Platform is using the Fixed Navigation mode and if the Control Button (0) is pressed, the control bit is toggled to allow the Robotic Platform to be deactivated. This deactivation stops the platform and deactivates the Fixed Navigation Mode. The deactivation of the Robotic Platform occurs when the Control bit = 0 and the software follows left path after the Control Variable is checked in Figure 12.1. This deactivation mode is also used on all of the manual modes of Stop / Reload Mode and Increment / Decrement Mode. At the flowchart there is a return, which means that both software modes return to the Check Control Variable in the Fixed Navigation Mode seen in Figure 19.1 in Appendix A.

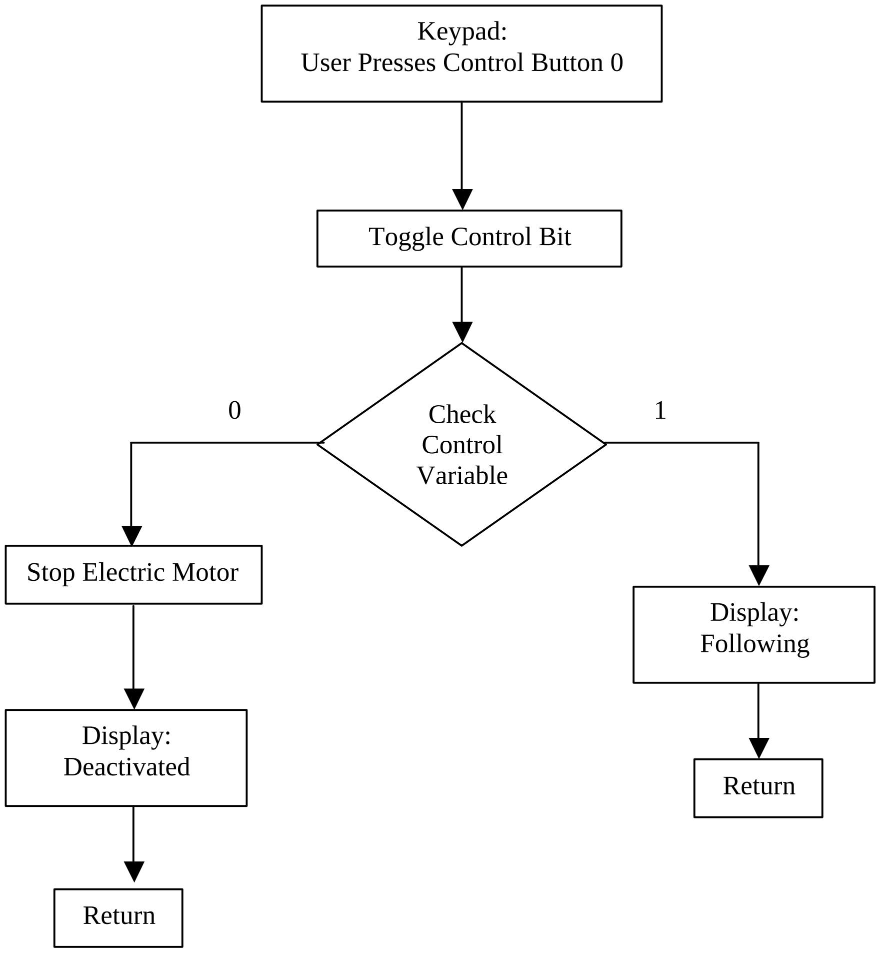


Figure 12.1
Navigation Control Mode Software Flowchart

Design Equations and Calculations

To calculate the PWM period from timer 2, the critical limiting factor will be the sensor subsystem. Using the timing diagram of the SRF04 seen in Appendix C1 the total period for the SRF04 was calculated in equation (13.1). The 10 μs corresponds to required trigger pulse. The 0.2 ms corresponds to the 40 KHz, 8 cycle sonic burst seen in the timing diagram Appendix C1. In lab the maximum pulse width measured corresponding to the largest measure distance from the SRF04 ultrasonic sensor was approximately 18 ms. 20 ms was chosen to allow for any errors in measure signals and also because the actual trigger pulse used is 1.5 ms. The 10 ms is needed for delay from the end of the echo pulse to the next trigger pulse. All these values resulted in equation (13.1) producing a 30.21 ms period signal for the SRF04, which is approximately 33 HZ.

$$\text{Total period for SRF04} = 10\text{us} + 0.2\text{ms} + 20\text{ms} + 10\text{ms} = 30.21\text{ms} \qquad (13.1)$$

The total timer counts equation (13.2) and the timer setting equation (13.3) are used to set the compare registers and reload values of timer 2.

$$\text{Total timer counts} = \frac{\text{Seconds}}{\frac{12}{\text{fosc}}} = \frac{\text{Seconds}}{\frac{12}{11.0592\text{ Mhz}}} \qquad (13.2)$$

$$\text{Timer setting} = 2^{16} - 1 - \text{Total Timer Counts} \qquad (13.3)$$

The timer 2 timer reload value setting produces a 30.21 ms period PWM signal which corresponds to approximately 33 Hz. Using equation (13.2) and equation (13.3) the Timer 2 initial setting is below.

Timer 2 count reload setting for 33 Hz PWM signal = 933Dh (13.4)

The timer 2 compare register 1 produces a 1.5 ms positive pulse width, which is used to set the steering subsystem in the neutral position and trigger the ultrasonic SRF04 sensor. Using equation (13.2) and equation (13.3) the timer 2 compare register 1 setup values are below.

Timer 2 Compare Register 1 Setting = FA99h (13.5)

Timer 2 compare register 2 will be used to produce a varying PWM signal to the electric motor subsystem. The three values that will be calculated will be for full reverse speed, neutral and full forward speed. The full reverse PWM signal corresponds to 0.7 ms, the neutral position PWM signal corresponds to 1.0 ms and full forward speed corresponds to 1.7 ms. The Timer 2 compare register 2 values are provided below for full reverse speed see (13.6), neutral see (13.7) and full forward speed see (13.8). These values were calculated using equations (13.2) and (13.3).

Timer 2 compare register 2 full reverse setting = FD7Ah (13.6)

Timer 2 compare register 2 neutral Setting = FC65h (13.7)

Timer 2 compare register 2 full forward setting = F8CCh (13.8)

Project Data

The EMAC microcontroller is set up using external interrupts to measure the PWM signal from the SRF04 ultrasonic sensor. On a rising edge of the PWM signal an interrupt occurs which resets timer 0 to zero and starts the timer. Upon the falling edge of the PWM signal another interrupt occurs which stops the timer. Using the Increment / Decrement Mode software, the increment unit was set to 1ms. The output of Timer 2 was connected to the pins of Timer 0. As the Timer 2 output values were changed from 1 ms to 18 ms, the Timer 0 values were recorded as seen in Table 14.1. The values in Table 14.1 were used to correct the measured PWM signal in software before the Fixed Navigation Mode. As can be seen in Table 14.1 the correction factor of positive 3 is added to the measured PWM signal from Timer 0.

Pulse Width (ms)	Calculated Hex Value	u Processor Measured Value	Timer count error
1	39Ah	397h	-3
2	734h	731h	-3
3	ACEh	ACBh	-3
4	E68h	E65h	-3
5	1202h	11FFh	-3
6	159Ch	1599h	-3
7	1936h	1933h	-3
8	1CD0h	1CCDh	-3
9	206Ah	2067h	-3
10	2404h	2401h	-3
11	279Eh	279Bh	-3
12	2B38h	2B35h	-3
13	2ED2h	2ECFh	-3
14	326Ch	3269h	-3
15	3606h	3603h	-3
16	39A0h	399Dh	-3
17	3D3Ah	3D37h	-3
18	40D4h	40D1h	-3

Table 14.1
Pulse width measurements measured by microcontroller.

After the SRF04 ultrasonic sensor was mounted, the hardware circuit was connected and some of the software was written. The SRF04 ultrasonic sensor distance in feet vs. measured hexadecimal values were shown in Table 15.1. This table shows how distance is related to the hexadecimal values measured from Timer 0. These data sets were used to set the desired distance in the software as seen in Table 15.2 based on which values were similar for the same distance.

	Measured Hex Value of Ultrasonic Sensor from EMAC		
Distance (ft)	**Set #1**	**Set #2**	**Set #3**
0.5	03C8h	03EAh	03EEh
1	06E8h	06FCh	06FCh
1.5	0A22h	0A3Eh	0A38h
2	0D44h	0D6Ah	0D5Ch
2.5	1088h	1082h	1088h
3	1392h	13B2h	13AAh
3.5	16D4h	16DEh	16DAh
4	19FCh	1A08h	1A00h
4.5	1D24h	1D3Ah	1D3Ch
5	2056h	204Ah	206Ah
5.5	2380h	239Eh	23A0h
6	26B6	26B8h	26C8
6.5	29EAh	2A02h	29FAh
7	2D02h	2D10h	2D2Ch
7.5	3042h	3060h	3056h
8	3368h	3374h	3376h
8.5	3686h	36CAh	36B8h
9	39DEh	39D4h	39F2h
9.5	3D04	3D12h	3D30h
10	402Ch	4048h	403Eh

Table 15.1
SRF04 ultrasonic sensor distance in feet vs. measured hex values from Timer 0

Distance (ft)	**Desired Distance Hex value**
1	06E8h
2	0D5Ah
3	13AAh
4	1A00h
5	2056h
6	26B8h
7	2D10h
8	3376h
9	39DEh

Table 15.2
Desired distance setting selected by the user and used for the Fixed Navigation Mode

Results / Conclusions

All the software modes were tested, debugged and worked correctly. The user is able to enter the fixed safety distance from 1 – 9 feet and select the desired out of range mode. The user is also able to activate or deactivate the navigation control mode and use the manual modes to increase, decrease, stop and reload the electric motor speed. The EMAC microcontroller triggers the ultrasonic sensor and measure the PWM echo signal, which is related to the distance the object is away from the ultrasonic sensor. The EMAC microcontroller is able to increase and decrease the speed of the electric motor to maintain the desired constant distance from the moving object in front of the robotic platform. A picture of the completed project is seen in Figure 16.1.

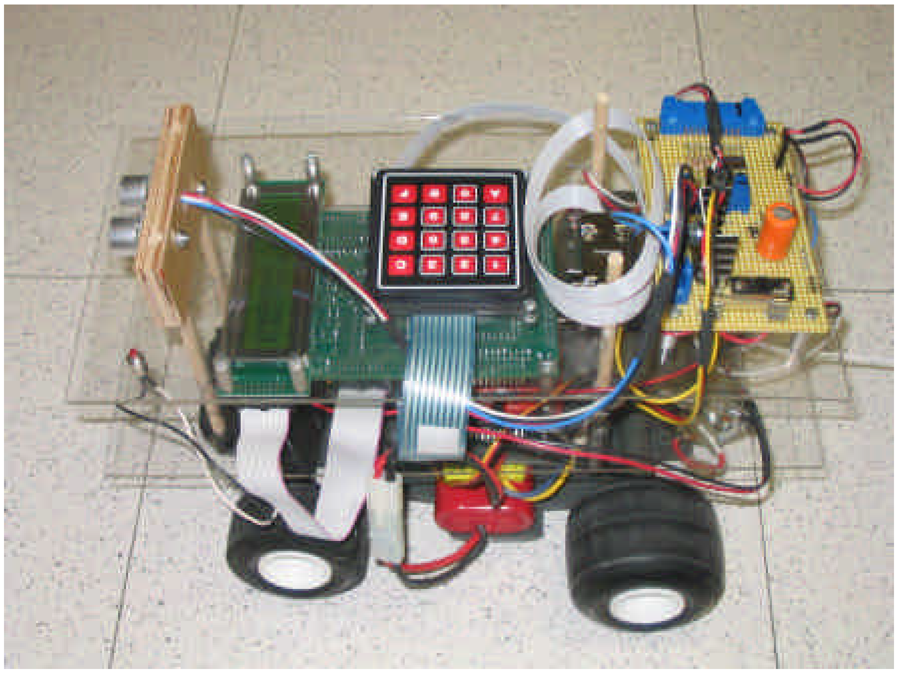


Figure 16.1
Completed Robotic Navigation Distance Control Platform

Future Development and Research of Project

Another expansion of this project might include modeling and determining a complete transfer function of the robotic platform. This would allow more advanced controls to be implemented into the project allowing the robotic platform to have smoother control, more accurate following distance and better tracking of the moving object.

A second expansion might include more sensors on the robotic platform to allow the robotic platform to be able to steer and navigate behind a moving object. Fuzzy logic steering control should be considered for this expansion due to the nonlinear steering required to steer the robotic platform around corners.

References

"Adaptive Cruise Control". I-Car Advantage. 16 February 2004. April 2004. <http://www.i-car.com/html_pages/about_icar/current_events_news/advantage/advantage_online_archives/2004/021604.html>

"Adaptive Cruise Control". April 2004. <http://www.ece.msstate.edu/classes/design/ece4532/2003_spring/cruise_control/>

"Adaptive Cruise Control Hits the Road". Automotive Industries. 1 October 1998. April 2004.<http://www.findarticles.com/cf_dls/m3012/1998_Oct_1/53179685/p1/article.jhtml>

"Announced Specification of HS-303 Standard sport Servo". Hitec. 18 November 2003 <www.hitecrcd.com/servos/discontinuedservos/hs303.pdf>

Clarke Peter. "Adaptive Cruise Control Takes to the Highway". EE Times. 20 October 1998. April 2004. <http://www.eetimes.com/story/OEG19981020S0007>

"ESC Specifications Reversible Models". Team Novak Electronics. 18 November 2003 <http://www.teamnovak.com/products/ESC_Specs/revers_spec/reverse_index.htm>

"General Servo Information". Hitec. 18 November 2003 <www.hitecrcd.com/support/manuals/servomanual.pdf>

Jones Willie D. "Keeping Cars from Crashing". September 2001. April 2004. <http://www.gavrila.net/Computer_Vision/Smart_Vehicles/Media_Coverage/spectrum.pdf>

"SRF04 Ultrasonic Ranger Technical Specifications". April 2004 <http://www.robot-electronics.co.uk/htm/srf04tech.htm>

"Rooster Operating Instructions". Team Novak Electronics,inc. 18 November 2003 <http://www.teamnovak.com/Download/acrobat/rooster_superr.pdf>

"Technical Info Glossary of Terms". Team Novak Electronics,inc. 18 November 2003 <http://www.teamnovak.com/Tech_info/glossary/index.html>

"Ultrasonic Rangers SRF04 & SRF08 FAQ". April 2004 <http://www.robot-electronics.co.uk/htm/sonar_faq.htm>

Appendix A:
Fixed Navigation Mode Flowchart

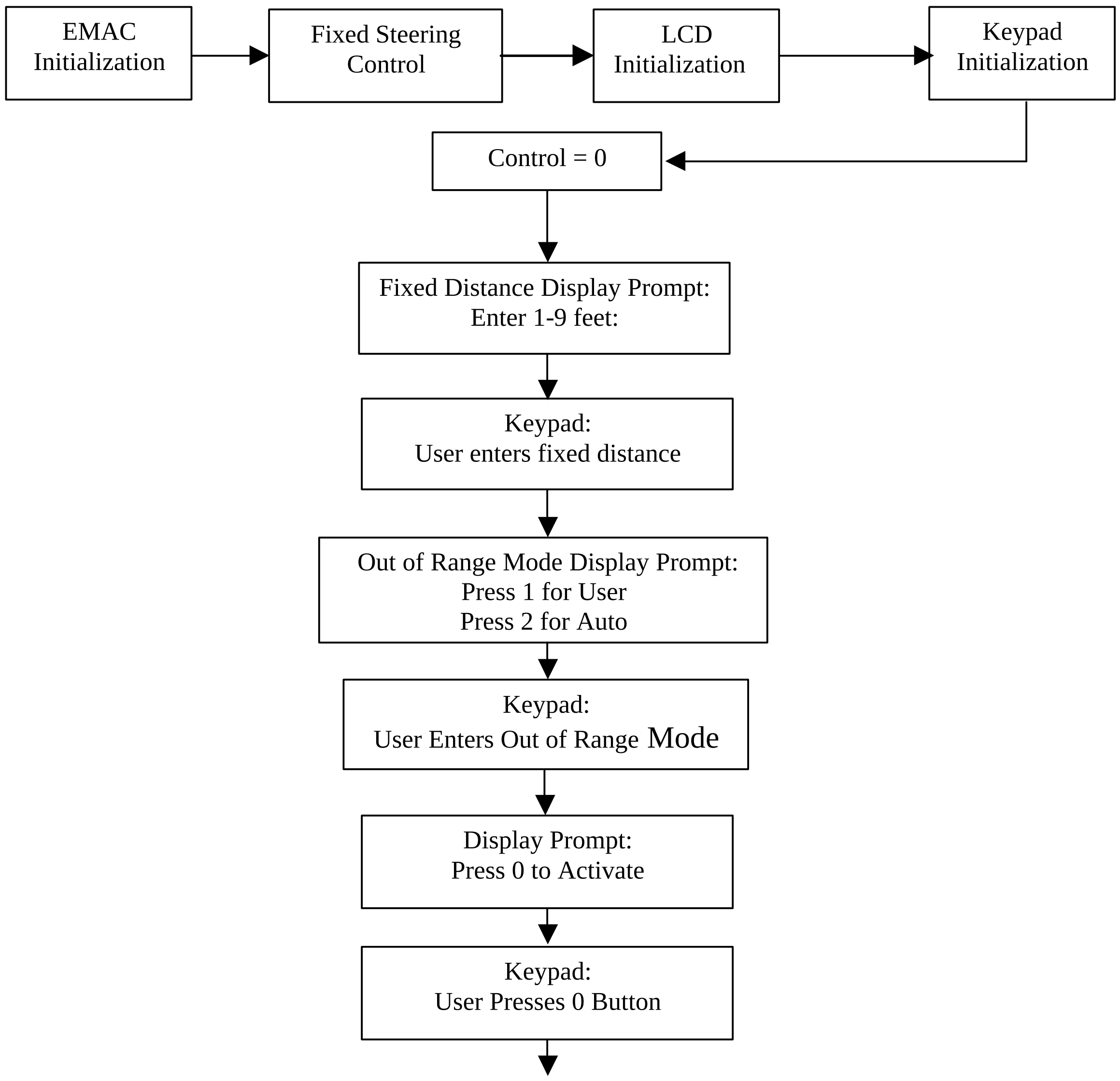

EMAC Initialization
Fixed Steering Control
LCD Initialization
Keypad Initialization
Control = 0
Fixed Distance Display Prompt: Enter 1-9 feet:
Keypad: User enters fixed distance
Out of Range Mode Display Prompt: Press 1 for User Press 2 for Auto
Keypad: User Enters Out of Range Mode
Display Prompt: Press 0 to Activate
Keypad: User Presses 0 Button

Figure 18.1
Fixed Navigation Mode Software Flowchart (1 of 2)

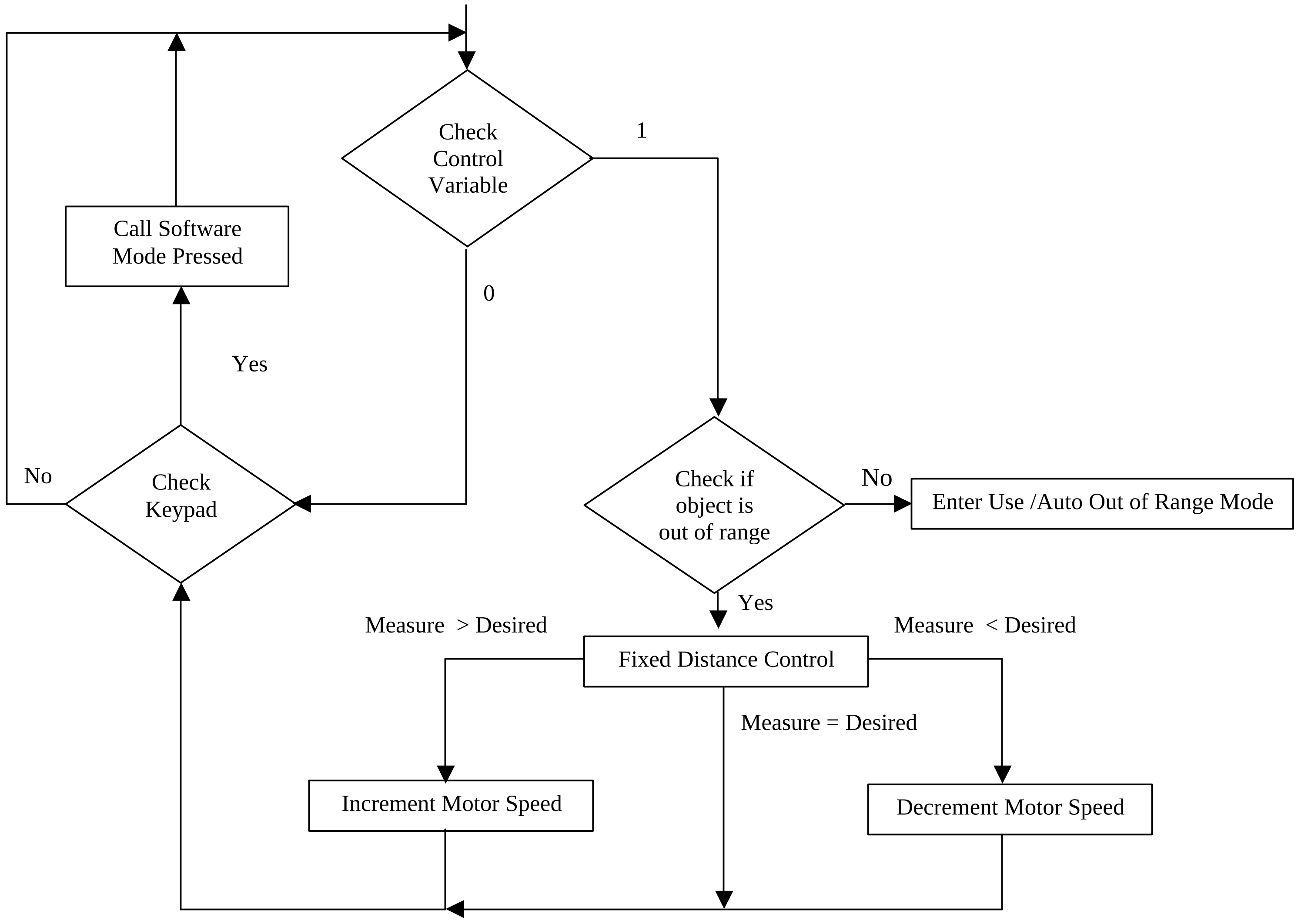


Figure 19.1
Fixed Navigation Mode Software Flowchart (2 of 2)

Appendix B1:
Header File Software (REG515.H)

Header file for 80515 microprocessor in C language

```
/*------------------------------------------------------------------------
REG515.H

Header file for 80515.
Copyright (c) 1988-1997 Keil Elektronik GmbH and Keil Software, Inc.
All rights reserved.
------------------------------------------------------------------------*/

/*  BYTE Register  */
sfr P0     = 0x80;
sfr P1     = 0x90;
sfr P2     = 0xA0;
sfr P3     = 0xB0;
sfr P4     = 0xE8;
sfr P5     = 0xF8;
sfr PSW    = 0xD0;
sfr ACC    = 0xE0;
sfr B      = 0xF0;
sfr SP     = 0x81;
sfr DPL    = 0x82;
sfr DPH    = 0x83;
sfr PCON   = 0x87;
sfr TCON   = 0x88;
sfr TMOD   = 0x89;
sfr TL0    = 0x8A;
sfr TL1    = 0x8B;
sfr TH0    = 0x8C;
sfr TH1    = 0x8D;
sfr SCON   = 0x98;
sfr SBUF   = 0x99;

sfr IEN0   = 0xA8;
sfr IEN1   = 0xB8;
sfr IP0    = 0xA9;
sfr IP1    = 0xB9;
sfr IRCON  = 0xC0;
sfr CCEN   = 0xC1;
sfr CCL1   = 0xC2;
sfr CCH1   = 0xC3;
sfr CCL2   = 0xC4;
sfr CCH2   = 0xC5;
sfr CCL3   = 0xC6;
sfr CCH3   = 0xC7;
sfr T2CON  = 0xC8;
sfr CRCL   = 0xCA;
sfr CRCH   = 0xCB;
sfr TL2    = 0xCC;
sfr TH2    = 0xCD;
sfr ADCON  = 0xD8;
sfr ADDAT  = 0xD9;
sfr DAPR   = 0xDA;

/*  BIT Register  */
/*  PSW  */
sbit CY     = 0xD7;
sbit AC     = 0xD6;
sbit F0     = 0xD5;
sbit RS1    = 0xD4;
sbit RS0    = 0xD3;
sbit OV     = 0xD2;
sbit F1     = 0xD1;
sbit P      = 0xD0;

/*  TCON  */
sbit TF1    = 0x8F;
sbit TR1    = 0x8E;
sbit TF0    = 0x8D;
sbit TR0    = 0x8C;
sbit IE1    = 0x8B;
sbit IT1    = 0x8A;
sbit IE0    = 0x89;
sbit IT0    = 0x88;

/*  IEN0  */
sbit EAL    = 0xAF;
sbit WDT    = 0xAE;
sbit ET2    = 0xAD;
sbit ES     = 0xAC;
sbit ET1    = 0xAB;
sbit EX1    = 0xAA;
sbit ET0    = 0xA9;
sbit EX0    = 0xA8;

/*  IEN1  */
sbit EXEN2  = 0xBF;
sbit SWDT   = 0xBE;
sbit EX6    = 0xBD;
sbit EX5    = 0xBC;
sbit EX4    = 0xBB;
sbit EX3    = 0xBA;
sbit EX2    = 0xB9;
sbit EADC   = 0xB8;

/*  P3  */
sbit RD     = 0xB7;
sbit WR     = 0xB6;
sbit T1     = 0xB5;
sbit T0     = 0xB4;
sbit INT1   = 0xB3;
sbit INT0   = 0xB2;
sbit TXD    = 0xB1;
sbit RXD    = 0xB0;

/*  SCON  */
sbit SM0    = 0x9F;
sbit SM1    = 0x9E;
sbit SM2    = 0x9D;
sbit REN    = 0x9C;
sbit TB8    = 0x9B;
sbit RB8    = 0x9A;
sbit TI     = 0x99;
sbit RI     = 0x98;

/*  T2CON  */
sbit T2PS   = 0xCF;
sbit I3FR   = 0xCE;
sbit I2FR   = 0xCD;
sbit T2R1   = 0xCC;
sbit T2R0   = 0xCB;
sbit T2CM   = 0xCA;
sbit T2I1   = 0xC9;
sbit T2I0   = 0xC8;

/*  ADCON  */
sbit BD     = 0xDF;
sbit CLK    = 0xDE;
sbit BSY    = 0xDC;
sbit ADM    = 0xDB;
sbit MX2    = 0xDA;
sbit MX1    = 0xD9;
sbit MX0    = 0xD8;

/*  IRCON  */
sbit EXF2   = 0xC7;
sbit TF2    = 0xC6;
sbit IEX6   = 0xC5;
sbit IEX5   = 0xC4;
sbit IEX4   = 0xC3;
sbit IEX3   = 0xC2;
sbit IEX2   = 0xC1;
sbit IADC   = 0xC0;
```

Appendix B2:
Keypad Scan Mode Software (KBDSCAN.a51)

Keypad scan mode software in assembly language

```
; KBD.a51
; Program file for keyboard in scanned mode
; This program was slightly modified by scott Sendra to work with C code
; Robotic Navigation Distance Control Platform
; Scott Sendra
; e-mail: ssendra@bradley.edu
; 5-11-04

Name chkkbd
kbdpt           equ     30h     ; value for P2 to access keyboard

kbd             SEGMENT         CODE
                RSEG    kbd
                PUBLICKBDinit
                PUBLICgetkbd
                PUBLICchkkbd
;
; initialize for keyboard scan
;
KBDinit:
                setb    it1             ; falling edge
                clr     ie1             ; clear flag
                clr     ex1         ; disable interrupt
                ret
;
; returns key value in A
;
getkbd:
                jnb     ie1,getkbd      ; WAIT FOR KEY
getkbd2:
                mov     p2,#kbdpt       ; point to keyboard
                movx    a,@r1
                anl     a,#00001111B    ; mask lower 4 bits
                add     a,#4;changed from 3 to 4 to keypad works with C code (Scott Sendra)
                movc    a,@a+pc; translate to character code
                mov     r7,a    ;was added so keypad works with C code (Scott Sendra)
                clr     ie1
                ret
; Keyboard conversion table
 db    '123C456D789EA0BF'
; returns key value in A if pressed
; returns 0 if not pressed
chkkbd:
                jb      ie1,getkbd2     ; CHECK FOR KEY
                mov     a,#0
                mov     r7,a    ;was added so keypad works with C code (Scott Sendra)
                ret
                end
```

Appendix B3
Keypad Interrupt Mode Software (KEYPAD.a51)

Keypad interrupt mode software in assembly language

```
;*************************************************************************
;
; KEYPAD subroutine: waits for key pressed and returns it
; in "key".
;
;*************************************************************************
$NOMOD51                        ; omit assembler micro definitions
$Include(reg515.inc)        ; define 515 micro

Name                        keypad
PUBLIC                      keypad,key_init

pad_key                     SEGMENT CODE
                            RSEG    pad_key ; switch to this code segment

                            USING  0                ; use register_bank 0


; Dempsey Note:
; This code was provided by EMAC
; It is not an efficient way to use keypad
; Normally must do other main code processing
;
; local definitions

KEYSEL                      EQU     38H     ; KEYPAD PORT

key_init:
                            SETB    IT1             ; falling edge trigger (INT1)
                            RET

keypad:
                            JNB     IE1,keypad      ; LOOP TILL KEY PRESSED
                            CLR     IE1             ; clear for next transition

                            PUSH    DPH
                            PUSH    DPL             ; SAVE DPTR
                            MOV     DPTR,#KEYTABL           ; POINT TO TRANSLATE TABLE
                            MOV     P2,#KEYSEL      ; POINT TO KEYPAD PORT
                            MOVX  A,@R1             ; GET KEY FROM PORT
                            ANL     A,#00001111B    ; ONLY 5 BITS
                            MOVC  A,@A+DPTR         ; TRANSLATE TO KEY FROM TABLE (ASCII)
                            MOV     R7,A            ; save in "R7" for C code
                            POP     DPL
                            POP     DPH
                            RET

KEYTABL: DB       '123C456D789EA0BF'

        END
```

Appendix B4:
LCD Initialization Software (LCD_GLD.a51)

LCD initialization software in assembly language

```
;*************************************************************************
;
; lcd_gld subroutine: modified lcddrv3.a51 from EMAC (see notes below)
;
;*************************************************************************

$NOMOD51                        ; omit assembler micro definitions
$Include(reg515.inc)        ; define 515 micro

Name                        lcd_gld
PUBLIC                      _lcdout,lcdinit

lcd_drv                     SEGMENT CODE
                            RSEG    lcd_drv  ; switch to this code segment

                            USING  0          ; use register_bank 0



;************************************************************************
;
; 20X2 character LCD drivers for the MICROPAC 535
;
; COPYRIGHT 1993-1995 EMAC INC.
;
; ESCflag is a bit field that must be declared.
; DLAYA is a 5ms delay routine with no registers affected.
; LCDINIT should be executed before any other LCD subroutines.
;
;
; LCDOUT: Output the char in ACC to LCD
; Following are some codes that are supported by this driver (20*2 displays)
;
;       0ah      move cursor to other line (i.e. from 1-2 or 2-1)
;       0dh      move cursor to beginning of line
;       1ah      clear display and move cursor to home
;       1bh      the next byte received after this will be written to register
;               0 of the lcd display
;

; definitions

escflag                     equu        psw.5     ; LCD equate
lcdcmd                      equ      28h          ; value for P2 to select lcd command port
initdata:
        db      38h,08,01,06,0eh,80h,0
;-------------------------------------------------------------
; Dempsey notes
; (1) R1,R7 are corrupted by this subroutine
; (2) Previously there were several returns- now only one: "LCDEXIT"
; (3) From calling program: Use a delay of 5ms (minimum)
;     between command codes LF (0A) and CR (0D)
;-------------------------------------------------------------

_lcdout:
                                                ; value passed in R& (C convention)
                            MOV    A,R7                     ; get character
                            MOV    P2,#LCDCMD               ; POINT TO COMMAND PORT
                            jnb    ESCflag,lcdnt5           ; skip if no ESC
                            clr    escflag
                            sjmp   reg0out                  ; write directly to lcd reg 0

lcdnt5:
                            ANL    A,#11100000B             ; SEE IF ANY OF UPPER 3 BITS SET
                            JNZ    REG1OUT                  ; IF YES, PRINT IT
                            MOV    A,R7                     ; RESTORE CHAR
                            ANL    A,#11111000B             ; SEE IF CHAR IS < 7
                            JZ     REG1OUT                  ; IF LESS, A=0 SO PRINT USER DEF CHAR 0-7

                            MOV    A,R7                     ; SEE IF CONTROL CHAR
                            CJNE   A,#0DH,LCNT1             ; IF NOT CR, SKIP
                            MOVX   A,@R1                    ; READ COMMAND PORT TO FIND CURSOR POS
                            SETB   ACC.7                    ; SET BIT 7 FOR DDRAM ADDR
                            ANL    A,#11100000 ; MOVE TO LEFT (ONLY VALID ON 2 LINE DISPL)
                            MOV    R7,A
                            SJMP   REG0OUT

LCNT1:          CJNE   A,#0AH,LCNT2             ; IF NOT LF, SKIP
                            MOVX   A,@R1                    ; READ COMMAND PORT TO FIND CURSOR POS
                            CPL    ACC.6                    ; SWITCH LINE (ONLY VALID ON 2 LINE DISPL)
                            SETB   ACC.7                    ; SET BIT 7 FOR DDRAM ADDR
                            MOV    R7,A
                            SJMP   REG0OUT

LCNT2:          CJNE   A,#1BH,LCNT3             ; IF NOT ESC, SKIP
                            setb   ESCflag                  ; indicate ESC received
                            JMP LCDEXIT

LCNT3:          CJNE   A,#1AH,LCNT4             ; EXIT IF NOT CLEAR SCREEN
                            MOV    R7,#1                    ; CLEAR COMMAND
                            SJMP   REG0OUT

                                                ; OUTPUT THE CHAR IN R2 TO REG 1
REG1OUT:
                            MOVX   A,@R1                    ; READ LCD COMMAND PORT
                            JB     ACC.7,REG1OUT            ; LOOP IF BUSY FLAG SET
                            INC    P2                       ; POINT TO LCD DATA PORT
                            MOV    A,R7                     ; RESTORE CHAR
                            MOVX   @R1,A                    ; OUTPUT IT
LCNT4:          JMP LCDEXIT


                                                ; OUTPUT THE CHAR IN R2 TO REG 0
REG0OUT:
                            MOVX   A,@R1                    ; READ LCD COMMAND PORT
                            JB     ACC.7,REG0OUT            ; LOOP IF BUSY FLAG SET
```

```
                        MOV     A,R7            ; RESTORE CHAR
                        MOVX  @R1,A             ; OUTPUT IT
                        JMP LCDEXIT

;
; LCDINIT: Init the LCD
;
LCDINIT:
                        clr       ESCflag             ; indicate no esc found
                        MOV     P2,#LCDCMD    ; POINT TO COMMAND PORT
                        LCALL  DLAYA            ; 5MS DELAY
                        LCALL  DLAYA            ; 5MS DELAY
                        LCALL  DLAYA            ; 5MS DELAY
                        LCALL  DLAYA            ; 5MS DELAY
                        MOV     A,#30H
                        MOVX  @R1,A           ; OUT TO LCD COMMAND PORT
                        LCALL  DLAYA            ; 5MS DELAY
                        MOVX  @R1,A           ; OUT TO LCD COMMAND PORT
                        LCALL  DLAYA            ; 5MS DELAY
                        MOVX  @R1,A           ; OUT TO LCD COMMAND PORT

                        MOV     DPTR,#INITDATA    ; POINT TO INIT DATA
                                        ; the last command should take no more than 40 uS.
                        mov      b,#80           ; for timeout of 80*3 * (12/clock)
lcdnit2:
                        movx     a,@r1                ; read lcd command port
                        jnb       acc.7,lcdnit1        ; exit if not busy
                        djnz      b,lcdnit2            ; loop till timeout
                        sjmp     lcdexit               ; exit if timeout

LCDNIT1:
                        MOVX  A,@R1               ; READ LCD COMMAND PORT
                        JB         ACC.7,LCDNIT1     ; LOOP IF BUSY FLAG SET

                        CLR      A
                        MOVC  A,@A+DPTR     ; GET BYTE FROM INIT TABLE
                        JZ         LCDEXIT          ; EXIT IF 0
                        INC      DPTR                ; POINT TO NEXT BYTE
                        MOVX  @R1,A              ; OUTPUT BYTE
                        SJMP    LCDNIT1           ; LOOP

LCDEXIT:
                        RET



;
; MISCELLANEOUS DELAYS
;
DLAYA:                  PUSH    ACC
                        MOV     A,#100
                        AJMP    DLAYA2

DLAYB:                  PUSH    ACC
                        MOV     A,#128
```

```
                AJMP    DLAYA2

DLAYC:          PUSH    ACC
                MOV     A,#255
                AJMP    DLAYA2
dlayd:          push    acc
                mov     a,#8


DLAYA2:
                PUSH    ACC
                MOV     A,#0FFH
DLAYA1:
                MOV     A,#0FFH
                DJNZ    ACC,$           ; LEVEL 3 LOOP
                POP     ACC
                DJNZ    ACC,DLAYA2      ; LEVEL 1 LOOP

                POP     ACC
                RET


                END
```

Appendix B5:
Startup Software (STARTUP.a51)

Startup software in EMAC microcontroller in assembly language

```
;------------------------------------------------------------------------------ ;  This file is part of the C51 Compiler
package
;  Copyright (c) 1995-1997 Keil Software, Inc.
;------------------------------------------------------------------------------
; Modified by G. Dempsey 7/11/00 for interrupts
; changed startup.a51 to absolute code starting at 8000h
; also required to locate at 8000h in linker options
;
;
;  STARTUP.A51:  This code is executed after processor reset.
;
;  To translate this file use A51 with the following invocation:
;
;     A51 STARTUP.A51
;
;  To link the modified STARTUP.OBJ file to your application use the following
;  BL51 invocation:
;
;     BL51 <your object file list>, STARTUP.OBJ <controls>
;
;------------------------------------------------------------------------------
$NOMOD51                                ; omit assembler micro definitions
$Include(reg515.inc)         ; define 515 micro
;
;  User-defined Power-On Initialization of Memory
;
;  With the following EQU statements the initialization of memory
;  at processor reset can be defined:
;
; the absolute start-address of IDATA memory is always 0
IDATALEN                EQU     080H    ; the length of IDATA memory in bytes.
;
XDATASTART              EQU     0H      ; the absolute start-address of XDATA memory
XDATALEN                EQU     0H      ; the length of XDATA memory in bytes.
;
PDATASTART              EQU     0H      ; the absolute start-address of PDATA memory
PDATALEN                EQU     0H      ; the length of PDATA memory in bytes.
;
;  Notes:  The IDATA space overlaps physically the DATA and BIT areas of the
;          8051 CPU. At minimum the memory space occupied from the C51
;          run-time routines must be set to zero.
;------------------------------------------------------------------------------
;
;  Reentrant Stack Initilization
;
;  The following EQU statements define the stack pointer for reentrant
;  functions and initialized it:
;
;  Stack Space for reentrant functions in the SMALL model.
```

```
IBPSTACK            EQU    1        ; set to 1 if small reentrant is used.
IBPSTACKTOP         EQU    0FFH+1 ; set top of stack to highest location+1.
;
;  Stack Space for reentrant functions in the LARGE model.
XBPSTACK            EQU    0        ; set to 1 if large reentrant is used.
XBPSTACKTOP         EQU    0FFFFH+1; set top of stack to highest location+1.
;
;  Stack Space for reentrant functions in the COMPACT model.
PBPSTACK            EQU    0        ; set to 1 if compact reentrant is used.
PBPSTACKTOP         EQU    0FFFFH+1; set top of stack to highest location+1.
;
;------------------------------------------------------------------------------
;
;  Page Definition for Using the Compact Model with 64 KByte xdata RAM
;
;  The following EQU statements define the xdata page used for pdata
;  variables. The EQU PPAGE must conform with the PPAGE control used
;  in the linker invocation.
;
PPAGEENABLE         EQU    0        ; set to 1 if pdata object are used.
PPAGE                      EQU    0        ; define PPAGE number.
;
;------------------------------------------------------------------------------

                           NAME   ?C_STARTUP

?STACK                     SEGMENT   IDATA

                           RSEG   ?STACK
                           DS     1

                           EXTRN CODE (?C_START)
                           PUBLIC?C_STARTUP


; Define starting location for program

stard               EQU    8000H           ; start address for program



                           CSEG   AT       stard

?C_STARTUP:         LJMP   STARTUP1        ; jump over interrupt vector table

;--------------------------------------------------------
; Interrupt Vector Table
; Area
;--------------------------------------------------------
                           CSEG   AT stard+0BH    ; 0BH=addr for Timer 0
                           LJMP   tmr0srv

                           CSEG   AT stard+13h    ; External interrupt 1.
                           LJMP   ext1srv

                           CSEG   AT stard+1BH    ; Timer 1 interrupt.
```

```
            LJMP    tmr1srv

            CSEG    AT stard+23H    ; Serial interrupt
            LJMP    serialsrv

            CSEG    AT stard+2BH    ; Timer 2
            LJMP    tmr2srv

            CSEG    AT stard+43H    ; IADC interrupt.
            LJMP    iadcsrv

            CSEG    AT stard+4BH    ; IEX2 interrupt.
        LJMP        iex2srv

            CSEG    AT stard+53H    ; IEX3 interrupt.
            LJMP    iex3srv

            CSEG    AT stard+5BH    ; IEX4 interrupt
            LJMP    iex4srv

            CSEG    AT stard+63H    ; IEX5 interrupt.
            LJMP    iex5srv

            CSEG    AT stard+6BH    ; IEX6 interrupt.
            LJMP    iex6srv


;----------------------------------------------------------
;
; Interrupt: Timer 0 Service
;
;----------------------------------------------------------
tmr0srv:                                ; timer 0 service
;------------------------------------------------------------
;----------------------------------------------------------
;
; Interrupt: Timer 2 Servie
;
;----------------------------------------------------------
tmr2srv:                                ; timer 2 service
;------------------------------------------------------------
;------------------------------------------------------------
;
; Unused Interrupt Service Routines
;
;------------------------------------------------------------
;
; note: this is not required but shown for completeness

ext1srv:            RETI                ; do nothing but return
tmr1srv:            RETI
serialsrv:          RETI
iadcsrv:            RETI

;----------------------------------------------------
```

```
                                        ;Scott Sendra added code
iex2srv:                      ;INT2 falling interrupt of PWM signal from distance sensor
clr TR0                                 ;stop timer 0
RETI
;---------------------------------------------------
                                        ;Scott Sendra added code
iex3srv:                      ;INT3 rising interrupt of PWM signal from distance sensor
mov TL0,#0                    ;resets timer 0 values
mov TH0,#0
setb TR0                      ;start timer 0
RETI
;---------------------------------------------------
iex4srv:                      RETI
iex5srv:                      RETI
iex6srv:                      RETI

;---------------------------------------------------------
;
; End of Interrupt Service Routines
;
;---------------------------------------------------------


STARTUP1:

; Initilization Specific To The EMAC MicroPac 535 SBC

                                        setb    P5.5            ; reset SC26C92 DUART
                                        clr     P5.5            ; bring DUART out of reset
                                        setb    P5.0            ; make A16 of 128K Ram, hi
                                        clr   P5.1              ; enable memory mapped IO
                                        clr     P5.2            ; disable EEPROM
; End Of MicroPac 535 Initilization

; Interrupt setup code
; local definitions

;DSEG AT 30h
;PWH:  ds 1
;PWL:  ds 1
;pwpw: ds 0Eh
;PUBLIC PWH
;PUBLIC PWL

CSEG

;mov PWH,#09h
;mov PWL,#19h

;Robotic Navigation Distance Control Platform code added
;Scott Sendra
;e-mail: ssendra@bradley.edu
;5-11-04
;software code added until setb TR0
```

```
mov CCL1,#0B7h           ;CC register 1 to produce 1.5ms pulse for servo
mov CCH1,#0FAh
mov CCEN,#28h            ;CC register 1 and 2 compared enabled
mov CCL2,#65h
mov CCH2,#0FCh
mov TH2,#93h             ;Places 933Dh into timer 2
mov TL2,#3Dh
mov CRCH,#93h            ;Timer 2 reload value of 933Dh
mov CRCL,#3Dh
mov a,T2CON              ;Timer 2 mode 0 with compared mode 0
mov a,#11h
mov T2CON,a

clr TR0                  ;stops timer 0
setb P1.0                ;enable INT3/P1.0 as input
setb P1.4                ;enable INT2/P1.4 as input
mov TL0,#0               ;Sets Timer 0 to 0 value
mov TH0,#0
mov a, TMOD              ; Timer 0 mode 1, TR0 control bit timer control
anl a,#0F0h
orl a,#01h
mov TMOD,a
setb I3FR                ;T2CON rising edge activated INT3/P1.0
clr I2FR                  ;T2CON falling edge activated INT2/P1.4
clr EX0                  ;Timer 0 external interrupt 0 disabled
setb IEN1.2              ;enable INT3
setb IEN1.1              ;enable INT2
setb EAL                 ;enable all interrupts
setb TR0                 ;Set timer 0 to start



IF IDATALEN <> 0
                                  MOV     R0,#IDATALEN - 1
                                  CLR     A
IDATALOOP:               MOV      @R0,A
                                  DJNZ    R0,IDATALOOP
ENDIF

IF XDATALEN <> 0
                                  MOV     DPTR,#XDATASTART
                                  MOV     R7,#LOW (XDATALEN)
  IF (LOW (XDATALEN)) <> 0
                                  MOV     R6,#(HIGH XDATALEN) +1
  ELSE
                                  MOV     R6,#HIGH (XDATALEN)
  ENDIF
                                  CLR     A
XDATALOOP:               MOVX     @DPTR,A
                                  INC     DPTR
                                  DJNZ    R7,XDATALOOP
                                  DJNZ    R6,XDATALOOP
ENDIF

IF PPAGEENABLE <> 0
                                  MOV     P2,#PPAGE
```

```
ENDIF

IF PDATALEN <> 0
                                MOV     R0,#PDATASTART
                                MOV     R7,#LOW (PDATALEN)
                                CLR     A
PDATALOOP:              MOVX    @R0,A
                                INC     R0
                                DJNZ    R7,PDATALOOP
ENDIF

IF IBPSTACK <> 0
EXTRN DATA (?C_IBP)

                                MOV     ?C_IBP,#LOW IBPSTACKTOP
ENDIF

IF XBPSTACK <> 0
EXTRN DATA (?C_XBP)

                                MOV     ?C_XBP,#HIGH XBPSTACKTOP
                                MOV     ?C_XBP+1,#LOW XBPSTACKTOP
ENDIF

IF PBPSTACK <> 0
EXTRN DATA (?C_PBP)
                                MOV     ?C_PBP,#LOW PBPSTACKTOP
ENDIF

                                MOV     SP,#?STACK-1
                                SETB    EAL             ; enable all interrupts
                                LJMP    ?C_START

                                END
```

Appendix B6:
Main Software (MAIN.C)

Main software for this project in C language

```
/*
Robotic Navigation Distance Control Platform Software
by Scott Sendra
e-mail: ssendra@bradley.edu
5-11-04
*/

#pragma SMALL
#include <reg515.h>

extern                  void lcdinit(void);
extern                  void key_init(void);
extern                  int lcdout(char dummy);
extern                  int keypad(void);
extern int chkkbd(void);
extern int KBDinit(void);

/****************************************
;
;       Main Module
;
;****************************************
*/

/* local definitions */

#define                 LCD_CLR     0x1A    // clear LCD command
#define                 LCD_LF      0x0A    // LCD line feed (next line)
#define                 LCD_CR      0x0D    // Move cursor to beginning of line
#define                 keyA    0x41    // key A
#define                 keyB    0x42    // key B
#define                 keyC    0x43    // key C
#define keyD            0x44    // key D
#define keyE            0x45  // key E
#define keyF            0x46    // key F
#define key0            0x30    // key 0
#define key1            0x31    // key 1
#define key2            0x32    // key 2
#define key3            0x33    // key 1
#define key4            0x34    // key 4
#define key5            0x35    // key 5
#define key6            0x36    // key 6
#define key7            0x37    // key 7
#define key8            0x38    // key 8
#define key9            0x39    // key 9
```

```
int i;                            //used in Display () to display arrays to LCD
int pw_low, pw_high;
int L2, H2;
int PW_CCL2, PW_CCH2; //Used in the stop mode to save the current compare register 2 values of current motor speed
int desiredPWH, desiredPWL;       //Used to store the desire distance from the user
int High, Low;                    //Used to store the high and low bytes from timer 0
int control;                      //Used to for the activate or deactive Navigation Mode
int SpeedLUnit, SpeedHUnit;       //Used to set the increment or decrement electric motor speed
int OutofRangeMode;               //Used to set store the User or Auto Out of Ranges Modes
int nop;                          //Used in for LCD display in several locations to creat delay
int count;                        //Used to control update of electric motor speed

/*All arrays below are used to display info to to the LCD.  The null '/' is used to terminate the display loop.
The maximum number of characters that can be displayed to the LCD is 20 characters.
*/

unsigned char data key;                                    // current key
unsigned char data lcd_char=LCD_CLR;                       // char for LCD
unsigned char code mainmenu1[21]="Enter 1-9 feet: /";      // the '/' char is used as a null char
unsigned char code mainmenu2[21]="or 0-7 for override/";
unsigned char code OutofRangeMenu[21]="Press 1 for User/";
unsigned char code OutofRangeMenu2[21]="Press 2 for Auto/";
unsigned char code OutofRangeDisplay[21]="Out of Range/";
//unsigned char code OutofRangeDisplay1[21]="Press 0 to activate/";
unsigned char code OutofRangeDisplay2[21]="Waiting for object/";
unsigned char code ManIncMotorSpeed[21]= "Manual Inc Speed/";
unsigned char code ManDecMotorSpeed[21]= "Manual Dec Speed/";
unsigned char code ManStopMotor[21]= "Manual Stop/";
unsigned char code ManLastSpeed[21]= "Reload Last Speed/";
unsigned char code ManFullMotorSpeed[21]= "Full Forward Speed/";
unsigned char code ManRevMotorSpeed[21]= "Full Reverse Speed/";
unsigned char code Following[21]= "Following/";
unsigned char code Deactivated[21]= "Deactivated/";
unsigned char code Activate[21]="Press 0 to activate/";

//Software functions
void LCDmainmenu();
void OutofRangeMainMenu();
void PulseWidthDisplay();
void HextoASCII (int PW);
void IncMotorSpeed();
void DecMotorSpeed();
void OutofRange();
void ActivateDisplay ();
void Display (unsigned char array[21]);


void main(void)             //Main loop of program
{
SpeedHUnit= 0x00;           //This is the high byte value to change the speed of the motor.
SpeedLUnit= 0x01;           //This is the low byte value to change the speed of the motor.
control= 0;                 //This sets control to zero which the Navigation Mode will be deactivated
OutofRangeMode=0;           //Initial OutofRangeMode which is no out of range mode.
lcdinit();                  //LCD initialization
//key_init();
```

```
KBDinit();                              /Keypad initializatoin
LCDmainmenu();//Allows user to enter the desired distance and then sets variables desiredPWL and desiredPWH
OutofRangeMainMenu();  //Allows user to enter in the Out of Range Mode
//PulseWidthDisplay();
count=255;                    //Was used experimently to slow down the motor update rate

while(1)
{

while (count > 0) {count= count - 1;}//Used to slow down the motor update rate, for smoother control
if ((TR0==0) && (control ==1))     //If Timer 0 is stopped and control = 1 then enter navigation mode
                              {
                              if (TH0 > 0x41)
//If pulse width from sensor is larger then 0x41 aprox 10 feet, then object is out of range

                              {   CCH2= 0xFC; //stop electric motor
                                 CCL2= 0x65;
                                        High=TH0;          //High will be used in the OutorRange() function
                                        OutofRange();
                              }

                              if (TL0 <= 0xFC)
                              //This adds 0x03 to the timer values to correct timer count error
                                        {
                                        Low = TL0+0x03; High= TH0;
                                        }
                              else
                                        {
                                        Low= (0x02 - (0xFF - TL0 ));  High= TH0 + 0x01;
                                        }
                              //PulseWidthDisplay();

   if ((desiredPWH - High) > 0) //If measure pulse width is larger then desired, then decrease motor speed
                                        {
                                        DecMotorSpeed();
                                        }
                              else if(((desiredPWL - Low) > 1) && (desiredPWH >= High))
                                        {
                                        DecMotorSpeed();
                                        }
                              else if ((desiredPWH - High) < 0 )
                              //if measure pulse width is smaller then desired, then icrease motor speed
                                        {
                                        IncMotorSpeed();
                                        }
                              else if (((desiredPWL - Low) < 1) && (desiredPWH <= High))
                                        {
                                        IncMotorSpeed();
                                        }

    count= 255;
                              }

key=0;
//key=keypad();               // call keypad subroutine
key=chkkbd();                 //Keypad in scan mode
```

```
if(key==keyA)                  // Motor full reverse
{
control= 0;                    //Deactivates the Navigation Mode.
_lcdout(LCD_CLR);
Display (ManRevMotorSpeed);    //Displays "Full Reverse Speed" on LCD
ActivateDisplay ();            //Displays "Press 0 to activate on the 2nd line of LCD
CCH2= 0xFD; CCL2= 0x7A;        //Set electric motor in full reverse speed.
}

else if(key==keyB)                          //Motor stop
{
control= 0;                                 //Deactivates the Navigation Mode.
_lcdout(LCD_CLR);
Display (ManStopMotor);                     //Displays "Manual Stop" on LCD
ActivateDisplay ();
PW_CCH2= CCH2;  PW_CCL2= CCL2;              //Saves current speed of electric motor
CCH2= 0xFC; CCL2= 0x45;                     //Stops electric motor
}

else if(key==keyC) // Increment motor speed
{
control= 0;                    //Deactivates the Navigation Mode
_lcdout(LCD_CLR);
Display (ManIncMotorSpeed);    //Displays "Manual Inc Speed" on LCD
ActivateDisplay ();
IncMotorSpeed();               //Calls increment motor speed function
}


else if(key==keyD)          // Start motor from last speed
{
control= 0;                 //Deactivates the Navigation Mode
_lcdout(LCD_CLR);
Display (ManLastSpeed);  //Display "Reload Last Speed" on LCD
ActivateDisplay ();
CCH2= PW_CCH2;           //Reloads the last speed when pressig the Motor stop button B
CCL2= PW_CCL2;
}

else if (key == keyE)          // Decrement motor speed
{
control= 0;                    //Deactivates the Navigation Mode
_lcdout(LCD_CLR);
Display (ManDecMotorSpeed);    //Displays "Manual Dec Speed" on LCD
ActivateDisplay ();
DecMotorSpeed();               //Calls decrement motor speed function
}

else if(key==keyF)   // Motor full forward
{
control= 0;                    //Deactivates the Navigation Mode
_lcdout(LCD_CLR);
Display (ManFullMotorSpeed);   //Displays "Full Forward Speed" on LCD
ActivateDisplay ();
CCH2= 0xF8; CCL2= 0xCC;        //Sets electric motor to full forward speed
}


else if                         (key==key1)
                                         {
                                /* This allows the user to reset the software so the fixed desired distance
                                as well as the Out of Range Mode can be changed.
                                */

                                         CCH2= 0xFC;                //Stops the electric motor
                                         CCL2= 0x45;
                                         control= 0;//Resets all the variables as in the begining of software code.
                                         OutofRangeMode=0;
                                         LCDmainmenu();
                                         OutofRangeMainMenu();

                                }

else if (key==key0)
                                   {
                                control = !control;
                                //toggles control bit to so navigation would be activated or deactivated
                                if (control == 1)   //If control = 1 then navigation mode is active
                                         {
                                         _lcdout (LCD_CLR);
                                         Display (Following); //Display "Following" on LCD
                                         }

      if (control == 0)         //If control = 0 then navigation mode is deactivated
                                {
                                                  CCH2= 0xFC;                //Stops electric motor
                                                  CCL2= 0x45;
                                _lcdout (LCD_CLR);
                                Display (Deactivated);      //Display "Deactivated" on LCD
                                                  ActivateDisplay ();                   //Displays "Press 0 to
activate" on 2nd of LCD
                                }
                                   }



}                                /* end while loop */
}                               /* end main loop */

/**********************************************************************/
/*
This function has a display prompt "Enter 1-9 feet:".  When the user enteres
the desired distance the desiredPWH, and desiredPWL values are set using the
values below.  These values are used in the main loop of the program.
*/

void LCDmainmenu()
{
                                _lcdout(LCD_CLR);         //clears LCD display
                                Display (mainmenu1);      //Displays "Enter 1-9 feet:"
```

```
//Key pad used in interrupt mode and waits for user to enter in the desired distance
key=keypad();
      if (key==key1) { _lcdout (key1); desiredPWH= 0x06; desiredPWL= 0xE8;}
else if (key==key2) { _lcdout (key2); desiredPWH= 0x0D; desiredPWL= 0x5A;}
else if (key==key3) { _lcdout (key3); desiredPWH= 0x13; desiredPWL= 0xAA;}
else if (key==key4) { _lcdout (key4); desiredPWH= 0x1A; desiredPWL= 0x00;}
else if (key==key5) { _lcdout (key5); desiredPWH= 0x20; desiredPWL= 0x56;}
else if (key==key6) { _lcdout (key6); desiredPWH= 0x26; desiredPWL= 0xB8;}
else if (key==key7) { _lcdout (key7); desiredPWH= 0x2D; desiredPWL= 0x10;}
else if (key==key8) { _lcdout (key8); desiredPWH= 0x33; desiredPWL= 0x76;}
else if (key==key9) { _lcdout (key9); desiredPWH= 0x39; desiredPWL= 0xDE;}

}

/**********************************************************************/
/*
This function asks the user to enter in the User out of range mode.
The LCD displays on line 1: "Press 1 for User" and on line 2: Press 2 for Auto".
The keypad is used in interrupt mode and waits for user to press a key.
The User Out of Range Mode is set to 1, and the default and also selectable
Auto OUt of Range Mode is set to 2.
*/

void OutofRangeMainMenu()
{
                    _lcdout(LCD_CLR);
                    Display(OutofRangeMenu);//Displays "Press 1 for User" on LCD

                    _lcdout(LCD_CR);
                    nop=0;
                    //The nop is used for delay purpose so LCD could display 2nd line correctly
                    nop=0;
                    _lcdout(LCD_LF);
                    Display(OutofRangeMenu2);     //Displays "Press 2 for Auto" on LCD

key=keypad();
if (key==key1) {OutofRangeMode=1;}     //OutofRangeMode variable is set here.
else {OutofRangeMode=2;}
_lcdout(LCD_CLR);
Display (Activate);                    //Displays "Press 0 to activate" on LCD
}

/*********************************************************************/
/*
This function is not used for the proper operation of the platform.
This was used for testing purposes, to display the timer 0 values on the LCD.
This function was used the the HextoACCII () function to display the 2 byte
values on the LCD screen.  For example to display the values of timer 0
on the LCD screen the following values will need to be used.
High= TH0;
Low= TL0;
*/

void PulseWidthDisplay()
{
_lcdout(LCD_CLR);
```

```
HextoASCII (High);
HextoASCII (Low);
}

/***********************************************************************/
/*
This function is currently not used for the correct operation or display
of the LCD screen.  This function was used to convert a 2 byte hex value to
ASCII to allow hex value to be displayed on the LCD screen.  This allow
the display the pulse width measurements from Timer 0 on the LCD screen.
This was needed to take data of distance vs. hex values from the
ultrasonic sensor.  To use this function for Timer 0 for example PW = TL0
which would provide the low byte of timer 0 value.
*/

void HextoASCII (int PW)
{
int H, L;
H=PW / 0x10;
L=PW - (H*0x10);
if(H >= 0x0A) H= (H-0x0A) + 0x11;
_lcdout(H + 0x30);
if (L >= 0x0A) L= (L-0x0A) + 0x11;
_lcdout(L + 0x30);
}

/***********************************************************************/
/*
This function is used in the Navigation Mode as well as the manual increment mode.
This increments the motor speed by the specified number of units set in the
SpeedLUnit and SpeedHUnit variables.  To increase limit forward motor speed the
values values in the if statement for the compare registers have to be decreased
to allow a higher motor speed.
*/

void IncMotorSpeed()
{
if ((CCH2 >= 0xFC) && (CCL2 >= 0x10)) //This limits the forward speed of the motor
{
if (CCL2 < SpeedLUnit)    //This accounts if a carry needs to be performed
                          {
                          CCL2= CCL2 - SpeedLUnit + 0xF0 + 0x10;
                          CCH2= CCH2 -  (SpeedHUnit + 0x01);
                          }
else
                          {
                          CCL2= CCL2 - SpeedLUnit; //increments motor speed by SpeedLUnit
                          CCH2= CCH2 - SpeedHUnit; //increments motor speed by SpeedHUnit
                          }
}
}
/***********************************************************************/
/*
This funciton is used in the Navigation Mode as well as the manual decrement mode.
This decrements the motor speed byt the specified number of units set in the
SpeedLUnit and SpeedHUnit variables.  If reversed is used then the values in
```

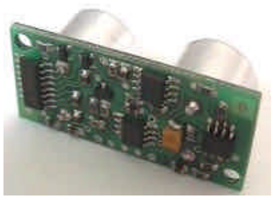
```
the if statement will need to be increased to allow the motor to reverse.
*/

void DecMotorSpeed()
{
if ((CCH2 <= 0xFC) && (CCL2 <= 0x45)) //This limits the Lower speed to prevent overflows during
decrementing.
{
/* for CCL2 0x45 was used because this was the largest pulse width that the ESC would accept to keep the
motor in netural.  The normal neutral setting is CCH2= 0xFC, CCL2= 0x65, but do to the resolution of the
ESC CCL2= 0x45 was the first  value when decreasing the pulse width that would put the motor in the
neutral position.  This higher neutral pulse width  would allow for a slightly faster response when
increasing the pulse to speed up the electric motor.
*/
                        if ((0xFF - CCL2) < SpeedLUnit) //This accounts if a carry needs to be
performed
                                {
                                CCL2= CCL2 + SpeedLUnit - 0xF0 - 0x10;
                                CCH2= CCH2 + (SpeedHUnit + 0x01);
                                }
                        else
                               {
                               CCL2= CCL2 + SpeedLUnit; //decrements motor speed by SpeedLUnit
                               CCH2= CCH2 + SpeedHUnit;//decrements motor speed by SpeedHUnit
                               }
}
else
{
                        CCL2= 0x45;             //This sets the Electric motor in the neutral position.
                        CCH2= 0xFC;
}
}
/***************************************************************/
/*
This function is the Out of Range Mode which is called from the main loop.
This displays "Out of Range" on the LCD and either follows the User or Auto
out of range flow charts.
*/

void OutofRange()
{

_lcdout(LCD_CLR);
Display(OutofRangeDisplay); //Displays the OutofRangeDisplay array

                        _lcdout(LCD_LF);
                        nop=0;
                        //The nop is used for delay purpose so LCD could display 2nd line correctly
                        nop=0;
                        _lcdout(LCD_CR);
                        if (OutofRangeMode == 1)
                        {   Display(Activate);

                                key=keypad();
                                //keypad in interrupt mode so waits for user to press a button.
                                if (key==key0)
```

```
                                        {

                                        }
                                while (TR0 != 0) {}
                                //once key is pressed it waits here until timer 0 is stopped
                        }
                        else
                        {   Display (OutofRangeDisplay2);  //Displays the OutofRangeDisplays2 array

                                while (High > 0x42)
                                {
                                /*Loops until the High value from the ultrasonic sensor is less
                                than 0x42 hex which is approx. 10 feet.  When the high value is
                                less than 0x42 then an object is with in 10 feet from the sensor.
                                */
                                        if (TR0 == 0) {High= TH0;}
                                }

                        }
_lcdout(LCD_CLR);
Display (Following);
}

/****************************************************************/
/*
This function is a function that displays a 20 char array of text
in a array on the LCD screen.  This function looks for the null char
'/' which loop of the array will stop.  Function accepts an array and
displays the char in the array until the null '/' char is reached.
*/

void Display (unsigned char array[21])
{
for (i=0; array[i] != '/'; i++)
                        _lcdout (array[i]);
}

/**************************************************************/
/*
This function is used to display for the manual modes:
increment / decrement, stop / reload, Navigaton control.
This displays "press 0 to activate" on the 2nd line of the LCD.
*/

void ActivateDisplay ()

{
_lcdout(LCD_CR);
nop=0;    //The nop is used for delay purpose so LCD could display 2nd line correctly
nop=0;
_lcdout(LCD_LF);
Display(Activate); //Displays Activate array
}
```

Appendix C1:
Data Sheet for SRF04 Ultrasonic Sensor

Source: http://www.robot-electronics.co.uk/htm/srf04tech.htm

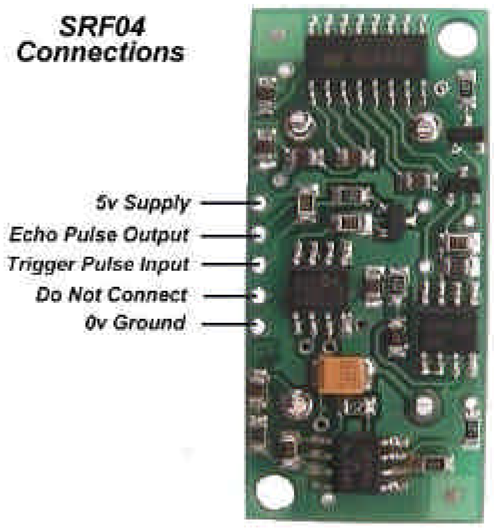
SRF04 - Ultra-Sonic Ranger
Technical Specification



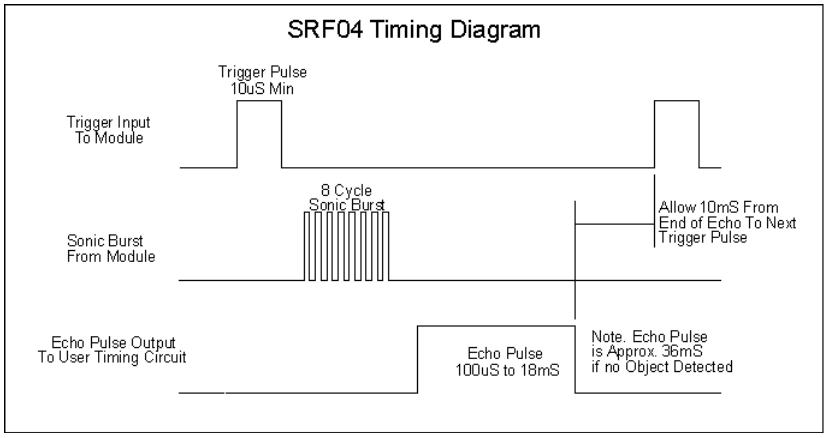
This project started after I looked at the Polaroid Ultrasonic Ranging module. It has a number of disadvantages for use in small robots etc.

1. The maximum range of 10.7 metre is far more than is normally required, and as a result
2. The current consumption, at 2.5 Amps during the sonic burst is truly horrendous.
3. The 150mA quiescent current is also far too high.
4. The minimum range of 26cm is useless. 1-2cm is more like it.
5. The module is quite large to fit into small systems, and
6. It's EXPENSIVE.

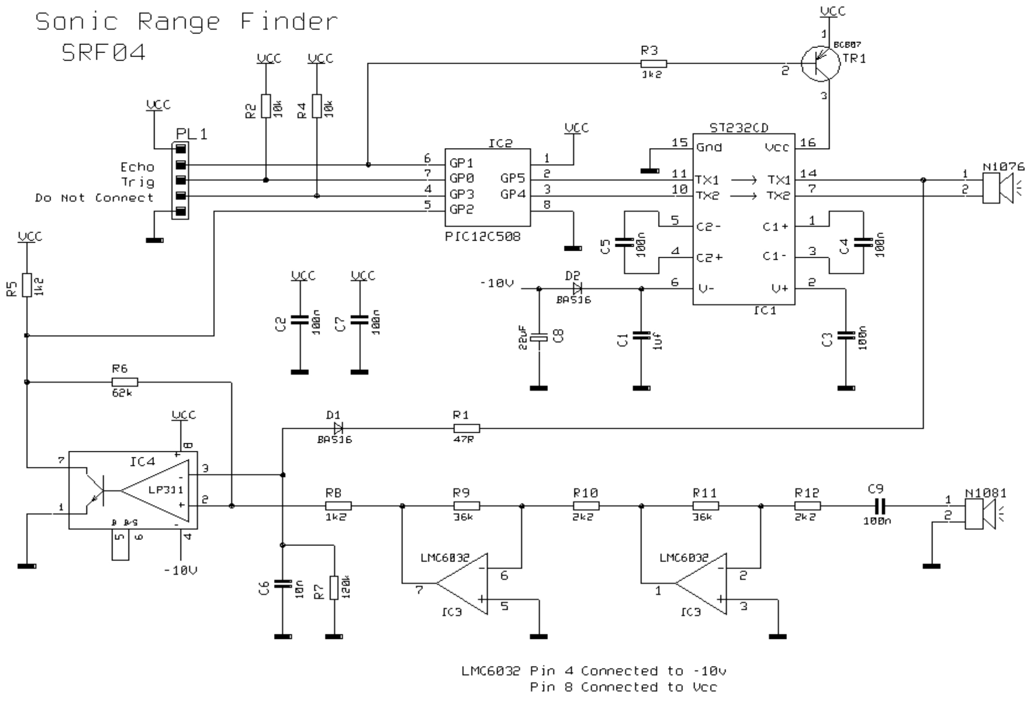
The SRF04 was designed to be just as easy to use as the Polaroid sonar, requiring a short trigger pulse and providing an echo pulse. Your controller only has to time the length of this pulse to find the range. The connections to the SRF04 are shown below:



The SRF04 Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging. The SRF04 will send out an 8 cycle burst of ultrasound at 40khz and raise its echo line high. It then listens for an echo, and as soon as it detects one it lowers the echo line again. The echo line is therefore a pulse whose width is proportional to the distance to the object. By timing the pulse it is possible to calculate the range in inches/centimeters or anything else. If nothing is detected then the SRF04 will lower its echo line anyway after about 36mS.


SRF04 Timing Diagram
Trigger Pulse 10uS Min
Trigger Input To Module
8 Cycle Sonic Burst
Allow 10mS From End of Echo To Next Trigger Pulse
Sonic Burst From Module
Echo Pulse Output To User Timing Circuit
Echo Pulse 100uS to 18mS
Note. Echo Pulse is Approx. 36mS if no Object Detected

Here is the schematic, You can download a better quality pdf (161k) version srf1.pdf



The circuit is designed to be low cost. It uses a PIC12C508 to perform the control functions and standard 40khz piezo transducers. The drive to the transmitting transducer could be simplest driven directly from the PIC. The 5v drive can give a useful range for large objects, but can be problematic detecting smaller objects. The transducer can handle 20v of drive, so I decided to get up close to this level. A MAX232 IC, usually used for RS232 communication makes and ideal driver, providing about 16v of drive.

The receiver is a classic two stage op-amp circuit. The input capacitor C8 blocks some residual DC which always seems to be present. Each gain stage is set to 24 for a total gain of 576-ish. This is close the 25 maximum gain available using the LM1458. The gain bandwidth product for the LM1458 is 1Mhz. The maximum gain at 40khz is 1000000/40000 = 25. The output of the amplifier is fed into an LM311 comparator. A small amount of positive feedback provides some hysterisis to give a clean stable output.

The problem of getting operation down to 1-2cm is that the receiver will pick up direct coupling from the transmitter, which is right next to it. To make matters worse the piezo transducer is a mechanical object that keeps resonating some time after the drive has been removed. Up to 1mS depending on when you decide it has stopped. It is much harder to tell the difference between this direct coupled ringing and a returning echo, which is why many designs, including the Polaroid module, simply blank out this period. Looking at the returning echo on an oscilloscope shows that it is much larger in magnitude at close quarters than the cross-coupled signal. I therefore adjust the detection threshold during this time so that only the echo is detectable. The 100n capacitor C10 is charged to about –6v during the burst. This discharges quite quickly through the 10k resistor R6 to restore sensitivity for more distant echo’s.

A convenient negative voltage for the op-amp and comparator is generated by the MAX232. Unfortunately, this also generates quite a bit of high frequency noise. I therefore shut it down whilst listening for the echo. The 10uF capacitor C9 holds the negative rail just long enough to do this.

In operation, the processor waits for an active low trigger pulse to come in. It then generates just eight cycles of 40khz. The echo line is then raised to signal the host processor to start timing. The raising of the echo line also shuts of the MAX232. After a while – no more than 10-12mS normally, the returning echo will be detected and the PIC will lower the echo line. The width of this pulse represents the flight time of the sonic burst. If no echo is detected then it will automatically time out after about 30mS (Its two times the WDT period of the PIC). Because the MAX232 is shut down during echo detection, you must wait at least 10mS between measurement cycles for the +/- 10v to recharge.

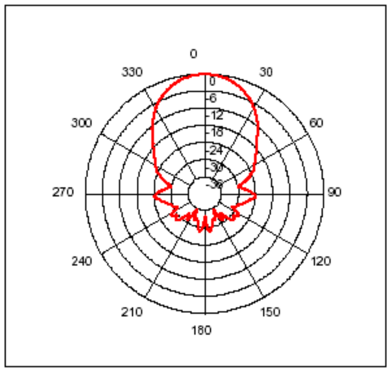
Performance of this design is, I think, quite good. It will reliably measure down to 3cm and will continue detecting down to 1cm or less but after 2-3cm the pulse width doesn't get any smaller.

Maximum range is a little over 3m. As and example of the sensitivity of this design, it will detect a 1inch thick plastic broom handle at 2.4m.
Average current consumption is reasonable at less than 50mA and typically about 30mA.

Download the source code and a ready assembled hex file.

Calculating the Distance
The SRF04 provides an echo pulse proportional to distance. If the width of the pulse is measured in uS, then dividing by 58 will give you the distance in cm, or dividing by 148 will give the distance in inches. uS/58=cm or uS/148=inches.

Changing beam pattern and beam width
You can't! This is a question which crops up regularly, however there is no easy way to reduce or change the beam width that I'm aware of. The beam pattern of the SRF04 is conical with the width of the beam being a function of the surface area of the transducers and is fixed. The beam pattern of the transducers used on the SRF04, taken from the manufacturers data sheet, is shown below.



There is more information in the sonar faq.

Update - May 2003
Since the original design of the SRF04 was published, there have been incremental improvements to improve performance and manufacturing reliability. The op-amp is now an LMC6032 and the comparator is an LP311. The 10uF capacitor is now 22uF and a few resistor values have been tweaked. These changes have happened over a period of time.

All SRF04's manufactured after May 2003 have new software implementing an optional timing control input using the "do not connect" pin. This connection is the PIC's Vpp line used to program the chip after assembly. After programming its just an unused input with a pull-up resistor. When left unconnected the SRF04 behaves exactly as it always has and is described above. When the "do not connect" pin is connected to ground (0v), the timing is changed slightly to allow the SRF04 to work with the slower controllers such as the Picaxe. The SRF04's "do not connect" pin now acts as a timing control. **This pin is pulled high by default and when left unconnected, the timing remains exactly as before.** With the timing pin pulled low (grounded) a 300uS delay is added between the end of the trigger pulse and transmitting the sonic burst. Since the echo output is not raised until the burst is completed, there is no change to the range timing, but the 300uS delay gives the Picaxe time to sort out which pin to look at and start doing so. The new code has shipped in all SRF04's since the end of April 2003. The new code is also useful when connecting the SRF04 to the slower Stamps such as the BS2. Although the SRF04 works with the BS2, the echo line needs to be connected to the lower numbered input pins. This is because the Stamps take progressively longer to look at the higher numbered pins and can miss the rising edge of the echo signal. In this case you can connect the "do not connect" pin to ground and give it an extra 300uS to get there.

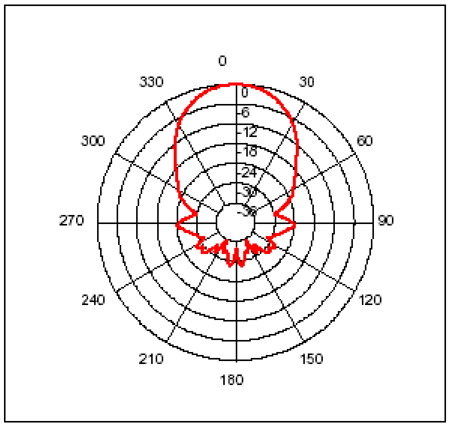
Appendix C2: FAQ for Ultrasonic SRF04 Sensor

Source: http://www.robot-electronics.co.uk/htm/sonar_faq.htm

Ultrasonic Rangers SRF04 & SRF08 FAQ

Q. What is the accuracy of the ranging?
A. We quote 3-4cm. Its normally better than this, however so many factors affect accuracy that we won't specify anything better than this. The speed of sound in air is approx. 346m/S at 24 degrees C. At 40KHz the wavelength is 8.65mm. The sonar's detect the echo by listening for the returning wavefronts. This echo has an attack/decay envelope, which means it builds up to a peak then fades away. Depending on which wavefront is the 1st to be strong enough to be detected, which could be the 1st, 2nd or even 3rd, the result can jitter by this much. Another effect which limits accuracy is a phasing effect where the echo is not coming from a point source. Take a wall for example, the ping will bounce off the wall and return to the sonar. The wall is large, however, and there will be reflections from a large area, with reflections from the outside being slightly behind the central reflection. It is the sum of all reflections which the sensor sees which can be either strengthened or weakened by phasing effects. If the echo is weakened then it may be the following wavefront which is detected - resulting in 8.65mm of jitter. It is possible to see changes of distance as small as mm but then get cm of jitter.

Q. How can I narrow the beam width?
A. You can't! This is a question which crops up regularly, however there is no easy way to reduce or change the beam width that I'm aware of. The beam pattern of the SRF04/8 is conical with the width of the beam being a function of the surface area of the transducers and is fixed. The beam pattern of the transducers used on the SRF04/8, taken from the manufacturers data sheet, is shown below.



Q. What are the units on the vertical axis in the beam pattern diagram?
A. Units are dB, taken from the manufacturers data sheet at: http://www.robot-electronics.co.uk/datasheets/t400s16.pdf

Q. What distance above the floor should the sonar be mounted?
A. If you can mount the SRF04/8 12in/300mm above the floor, that should be OK. If you mount them lower, you may need to point them upwards slightly to avoid reflections from the carpet pile or ridges in a concrete floor.

Q. Can we replace the transducers with sealed weatherproof types?
A. No. We have tried these on both the SRF04 and SRF08 and they do not work. The characteristics of the sealed devices requires a new design which is on our future plans list.

Q. What is the RH limit for the transducers?
A. This is not specified by the transducer manufacturers and is not listed in the data sheet. The following is the manufacturers response to an email "The RH here in Taiwan is normally higher than 95%. Just if this sensor(400ST/R160) is used in the air, it should be okay. Don't use in outdoors. Exposing in rainy day or underwater is not allowed."

Q. Is there a need for us to change the SRF08 address when using the sensor, can't I just use the default address?
A. Yes, if you only have one sensor you can use the default shipped address of 0xE0. You only need to set addresses if you are using more than one SRF08 on the same I2C bus.

Q. Can I fire two or more sonar's at the same time?
A. No! If two or more sonar's are fired together then they could pick up each other "ping" resulting in a false readings. Fire them sequentially 65mS apart

A. Yes! We do this all the time on our test robot, firing 8 SRF08's at the same time. They are facing outwards and fitted around a 15inch diameter circle. The gain is set to minimum and they are fired using the I2C general call at address 0, and read individually at their set addresses. Under these circumstances there is no direct interference.
A. Possibly! - Try it, and compare the results with firing them sequentially at 65mS intervals..

Q. If I change the SRF08 I2C address, will it stay at that address next time I switch on or do I need to set it every time?
A. You only need to set it once and it stays set to the new address - even when you power up again. The I2C address is stored in EEPROM and stays the same until you deliberately change it.

Q. If I change the SRF08 Range and Gain registers, will they stay the same the next time I switch on or do I need to set them every time?
A. Unlike the address, which is permanent, You will need to set the Range and Gain when you power up again.

Q. Can I change the sonar frequency of 40KHz to something else?
A. No. The frequency must be 40KHz, because that is the only frequency the transducers will operate at. Also the circuitry is designed to operate at 40KHz so you cannot change the transducers to other frequency types.

Q. If I reduce the range setting of the SRF08, can I fire the sonar faster?
A. Yes, but be careful. If you fire the sonar and there is nothing in the immediate range, than on the second firing, you may pick up an echo of the first ping which has only just arrived from a distant object. The second ranging will falsely interpret this as an echo from a nearby object. To avoid this, don't fire the sonar more frequently than every 60mS or so.

Appendix D1:
Specifications for Team Novak Rooster ESC

Source: http://www.teamnovak.com/products/ESC_Specs/revers_spec/reverse_index.htm

REVERSIBLE MODELS (CURRENT)

SPEC / FEATURE	ROOSTER
Part Number	#1850
List Price	$139.00
Input Voltage (cells)	7-Jun
Case Size (in)	1.63 x 2.02 x 1.22
Case Size (cm)	4.14 x 5.13 x 3.10
Weight (ounces/grams)	3.0 / 85.0
On-Resistance* (ohms)	0.018
Motor Limit	15 turns (at 6 cells)
One-Touch Set-Up	Yes
Drive Frequency (Hz)	1250
Brake Frequency (Hz)	1250
Discrete Steps	64: 32 Forward, 32 Reverse
Rated Fwd. Current* (Amps)	100
Rated Rev. Current* (Amps)	100
Braking Current* (Amps)	100
B.E.C. (volts / amps)	5.7 / 0.5
Wire Size (gauge)	16
Polar Drive Circuitry	Yes
Radio Priority Circuitry	Yes
Digital Anti-Glitch Circuitry	Yes
Reverse Voltage Protection	No
Thermal Protection	Yes: Dual-Level
Reverse Disable	Yes
Smart Braking Circuitry	Yes
Heat Sinks	Factory Installed
Brake Light Circuitry	Yes
Brake Light Kit	Optional
Battery Plug Installed	Tamiya
Motor Plug Installed	Bullet-Style

Appendix D2:
Definitions of Team Novak Rooster ESC

Source: http://www.teamnovak.com/Tech_info/glossary/index.html

Note: Definitions below from source website.

BEC:
The Abbreviation for *Battery Elimination Circuitry.* The BEC is a built-in voltage regulator that supplies a constant voltage to the receiver and servo.

Brake Light Circuitry:
The tiny circuit in the ESC that allows high intensity red LEDs to be illuminated when the ESC is in neutral or brakes. The LEDs are attached to the vehicle to function as brake lights. A Brake Light Kit is available from Novak (#5655).

Brake PWM Frequency:
The frequency at which the duty cycle information is being sent from the speed control to the motor for braking. It also controls the deceleration characteristics of your vehicle with respect to trigger movement in the Full Brake direction. Brake PWM Frequency is measured in Hertz (Hz).

Braking Current:
The amount of force or power the brake circuit can deliver; usually the more the better. ESC's with higher braking currents can provide better braking without fading.

Digital Anti-Glitch Circuitry:
An exclusive feature from Novak that rejects signals read by the speed control from the receiver that are caused by radio interference.

Discrete Steps:
The smallest motion change that can be distinguished from neutral to full throttle. The more steps a speed control uses to accelerate (or decelerate), the smoother the driving will be. Most racing ESCs have 64 steps, but the Novak Atom or Cyclone has 256 steps to create the smoothest trigger response available.

Drive PWM Frequency:
The frequency at which the duty cycle information is being sent from the speed control to the motor during forward drive (How many times-per-second the motor is being cycled on and off to control its speed). It also controls the acceleration characteristics of your vehicle with respect to trigger movement in the Full Throttle direction. Drive PWM Frequency is measured in Hertz (Hz).

Input Voltage:
The minimum or maximum voltage in which the ESC is designed to operate. To obtain Input Voltage, multiply the number of cells by 1.2 volts. For example, when we specify that the ESC will work from 4-10 cells, the input voltage is 4.8 to 12.0 volts.

Motor Limit:
A guideline for the lowest recommended number of turns that can be used with a particular ESC. The turns in a motor are the number of windings on the armature of the motor. The lower the number of turns, the lower the resistance of the motor. This lowered resistance results in a potentially higher current draw, which can cause the ESC to run hotter. Our motor limits are based on using a single motor in 1/10th scale vehicle, with 6-cells, and a gear ratio of 4:1 or higher. Your gearing, driving style, number of cells, tire size, ambient temperature, vehicle weight, and the amount of air flow over the heat sinks will effect the amount of heat build-up in the ESC, motor, and batteries. If you use a motor with fewer than the recommended minimum number of turns, you will need to monitor the ESC for excessive heat. If you use more than 6-cells you will need to increase the number of turns on the motor to prevent damage to your ESC. For dual motor recommendations, see the "Wire Dual Motors" page in our "How To..." section.

On-Resistance:
The restriction an ESC offers to the flow of the current to the motor at full speed. The lower the on-resistance, the higher the efficiency (performance) of the ESC. We measure the ESC's on-resistance based on the transistor's rating at 25 degrees Celsius junction temperature. For example, our Cyclone uses 6 HYPERFET III transistors in parallel that are rated by the manufacturer at 0.004 ohms each. To determine the Cyclone's total on-resistance, we use the following formula: [Transistor Rating] / [Number of Transistors] = [0.004 ohms] / [6 transistors] = 0.00067 ohms.

One-Touch Set-Up:
One-Touch Set-Up: A Novak first! Our One-Touch system allows the user to automatically adjust the speed control to the transmitter with the touch of a button. This system eliminates the need for manual transmitter adjustments using the neutral and high speed pots. In our Cyclone and Atom ESCs, the One-Touch button is also used to select a driving profile.

Polar Drive Circuitry:
A Novak exclusive feature which allows the circuitry to stay cool while enabling the speed control to handle higher powered motors. The results include a smoother performance, increased acceleration, longer run time and increased radio system range.

Radio Priority Circuitry:
When battery power is running low, this circuitry makes sure that power keeps being sent to the receiver. This maintains control of the model, even after the batteries have discharged.

Rated Current:
Rated current, or peak current, is the MOSFET's ability to handle high current surges for a very short duration (1-2 microseconds).

Reverse Disable:

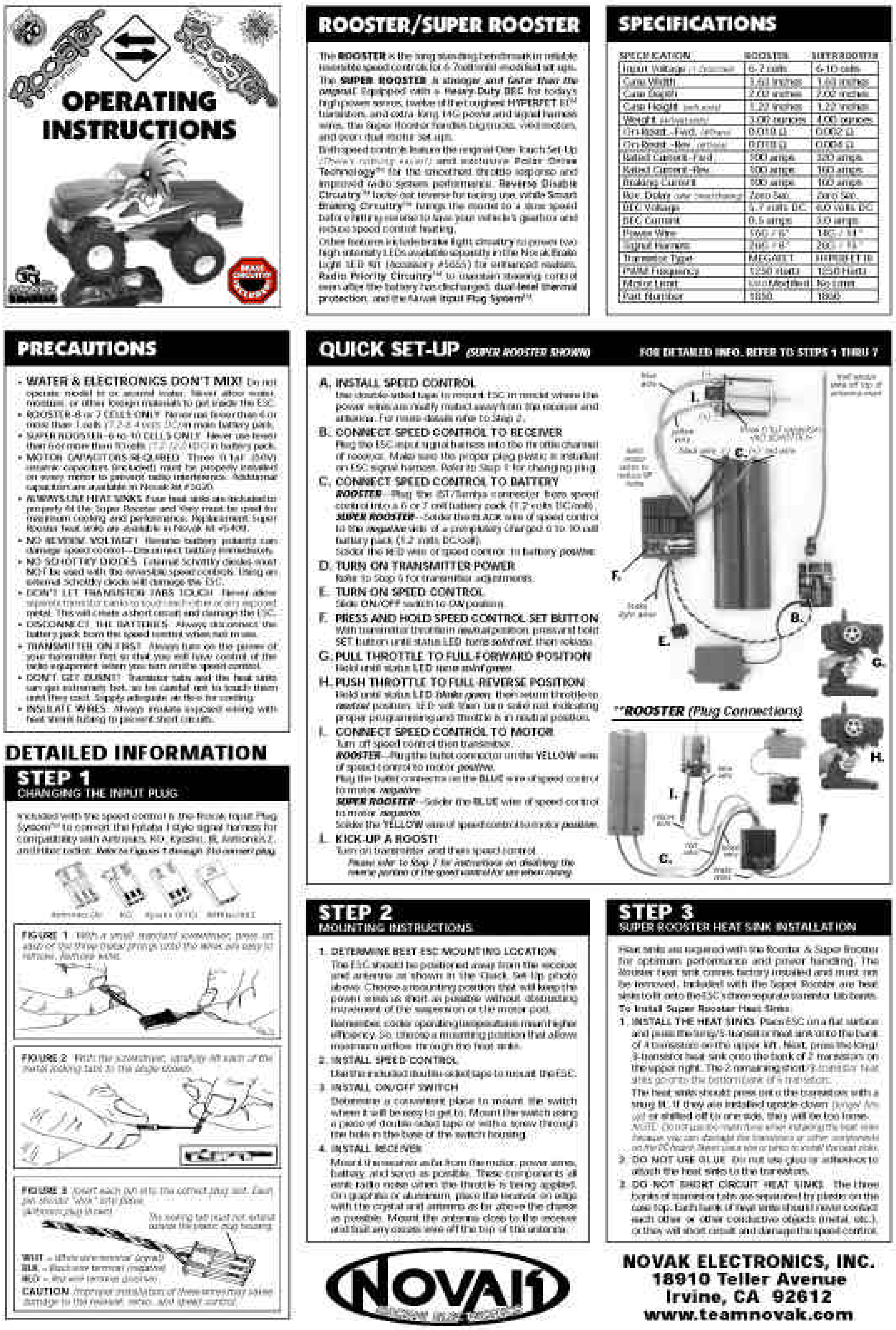
A feature in all currently manufactured Novak Reversible Electronic Speed Controls which enables a driver to turn off or "lock out" reverse for racing situations. When reverse is disabled, the ESC operates as a forward-only speed control with brakes. Reverse Disabling is turned on and off using the One-Touch Set-Up button.

Reverse Voltage:
When the power source (battery pack) is connected backwards to the ESC's red and black wires.

Smart Braking:
A feature in all currently manufactured Novak Reversible Electronic Speed Controls which is designed to help reduce wear and tear on the model's drive train and also reduce the amount of heat build up in the ESC. When reverse throttle is applied (while the model is moving forward), the Smart Braking Circuitry will apply brakes until the vehicle is moving at a slow enough speed where damage and excessive heat are not likely to occur. When the model slows to a safe speed, the Smart Braking Circuitry will then allow reverse to engage. Smart Braking only occurs when the ESC's reverse is enabled (see definition on Reverse Disable), the vehicle is moving, and reverse throttle is applied.

Thermal Overload Protection (Also referred to as Thermal Protection):
Thermal Overload Protection is a built-in sensor, which shuts down the MOSFET(s) when its temperature exceeds a preset level. This circuitry provides protection from overloads. Dual-level protection cuts the throttle in half when the ESC temperature reaches unsafe levels. If the temperature continues to climb, it will shut down.

Appendix D3:
Operation Instructions for Team Novak Rooster ESC
Source: http://www.teamnovak.com/Download/acrobat/rooster_superr.pdf


Rooster
OPERATING INSTRUCTIONS
Rooster

ROOSTER/SUPER ROOSTER

The **ROOSTER** is the long standing benchmark in reliable reversible speed controls for 6-7cell [illegible] modified set ups.

The **SUPER ROOSTER** is stronger and faster than the original. Equipped with a Heavy-Duty BEC for today's high power servos, twelve of the toughest HYPERFET III™ transistors, and extra long 14G power and signal harness wires, the Super Rooster handles big trucks, wild motors, and even dual motor set ups.

Both speed controls feature the original One Touch Set-Up (There's nothing easier!) and exclusive Polar Drive Technology™ for the smoothest throttle response and improved radio system performance. Reverse Disable Circuitry™ locks out reverse for racing use, while Smart Braking Circuitry™ brings the model to a slow speed before letting reverse to save your vehicle's gearbox and reduce speed control heating.

Other features include brake light circuitry to power two high intensity LEDs available separately in the Novak Brake Light LED Kit (Accessory #5655) for enhanced realism, Radio Priority Circuitry™ to maintain steering control even after the battery has discharged, dual-level thermal protection, and the Novak Input Plug System™.

SPECIFICATIONS

SPECIFICATION	ROOSTER	SUPER ROOSTER
Input Voltage (1.2v/cell)	6-7 cells	6-10 cells
Case Width	1.63 inches	1.63 inches
Case Depth	2.02 inches	2.02 inches
Case Height (with [illegible])	1.22 inches	1.22 inches
Weight (w/o [illegible])	3.00 ounces	4.00 ounces
On-Resist.–Fwd. (@Trans)	0.010 Ω	0.002 Ω
On-Resist.–Rev. (@Trans)	0.010 Ω	0.004 Ω
Rated Current–Fwd	300 amps	320 amps
Rated Current–Rev	300 amps	160 amps
Braking Current	300 amps	160 amps
Rev. Delay (after [illegible])	Zero Sec.	Zero Sec.
BEC Voltage	5.7 volts DC	6.0 volts DC
BEC Current	0.5 amps	3.0 amps
Power Wire	16G / 6"	14G / 14"
Signal Harness	26G / 8"	26G / 14"
Transistor Type	MEGAFET	HYPERFET III
PWM Frequency	1250 Hertz	1250 Hertz
Motor Limit	[illegible] Modified	No Limit
Part Number	1830	1890

PRECAUTIONS

- WATER & ELECTRONICS DON'T MIX! Do not operate model in or around water. Never allow water, moisture, or other foreign materials to get inside the ESC.
- ROOSTER–6 or 7 CELLS ONLY. Never use fewer than 6 or more than 7 cells (7.2-8.4 volts DC) in main battery pack.
- SUPER ROOSTER–6 to 10 CELLS ONLY. Never use fewer than 6 or more than 10 cells (7.2-12.0 VDC) in battery pack.
- MOTOR CAPACITORS REQUIRED. Three 0.1µF (50V) ceramic capacitors (included) must be properly installed on every motor to prevent radio interference. Additional capacitors are available in Novak kit #5620.
- ALWAYS USE HEAT SINKS. Four heat sinks are included to properly fit the Super Rooster and they must be used for maximum cooling and performance. Replacement Super Rooster heat sinks are available in Novak kit #5408.
- NO REVERSE VOLTAGE! Reverse battery polarity can damage speed control—Disconnect battery immediately.
- NO SCHOTTKY DIODES. External Schottky diodes must NOT be used with the reversible speed controls. Using an external Schottky diode will damage the ESC.
- DON'T LET TRANSISTOR TABS TOUCH. Never allow separate transistor banks to touch each other or any exposed metal. This will create a short circuit and damage the ESC.
- DISCONNECT THE BATTERIES. Always disconnect the battery pack from the speed control when not in use.
- TRANSMITTER ON FIRST. Always turn on the power of your transmitter first so that you will have control of the radio equipment when you turn on the speed control.
- DON'T GET BURNT! Transistor tabs and the heat sinks can get extremely hot, so be careful not to touch them until they cool. Supply adequate air flow for cooling.
- INSULATE WIRES. Always insulate exposed wiring with heat shrink tubing to prevent short circuits.

QUICK SET-UP *(SUPER ROOSTER SHOWN)* FOR DETAILED INFO, REFER TO STEPS 1 THRU 7

A. INSTALL SPEED CONTROL
Use double-sided tape to mount ESC in model where the power wires are readily routed away from the receiver and antenna. For more details refer to Step 2.

B. CONNECT SPEED CONTROL TO RECEIVER
Plug the ESC input signal harness into the throttle channel of receiver. Make sure the proper plug plastic is installed on ESC signal harness. Refer to Step 1 for changing plug.

C. CONNECT SPEED CONTROL TO BATTERY
ROOSTER—Plug the IST/Tamiya connector from speed control into a 6 or 7 cell battery pack (1.2 volts DC/cell).
SUPER ROOSTER—Solder the BLACK wire of speed control to the *negative* side of a completely charged 6 to 10 cell battery pack (1.2 volts DC/cell).
Solder the RED wire of speed control to battery *positive*.

D. TURN ON TRANSMITTER POWER
Refer to Step 5 for transmitter adjustments.

E. TURN ON SPEED CONTROL
Slide ON/OFF switch to ON position.

F. PRESS AND HOLD SPEED CONTROL SET BUTTON
With transmitter throttle in neutral position, press and hold SET button until status LED turns *solid red*, then release.

G. PULL THROTTLE TO FULL-FORWARD POSITION
Hold until status LED turns *solid green*.

H. PUSH THROTTLE TO FULL-REVERSE POSITION
Hold until status LED *blinks green*, then return throttle to *neutral* position. LED will then turn solid red indicating proper programming and throttle is in neutral position.

I. CONNECT SPEED CONTROL TO MOTOR
Turn off speed control then transmitter.
ROOSTER—Plug the bullet connector on the YELLOW wire of speed control to motor *positive*.
Plug the bullet connector on the BLUE wire of speed control to motor *negative*.
SUPER ROOSTER—Solder the BLUE wire of speed control to motor *negative*.
Solder the YELLOW wire of speed control to motor *positive*.

J. KICK-UP A ROOST!
Turn on transmitter and then speed control.
Please refer to Step 7 for instructions on disabling the reverse portion of the speed control for use when racing.

DETAILED INFORMATION

STEP 1
CHANGING THE INPUT PLUG

Included with the speed control is the Novak Input Plug System™ to convert the Futaba J style signal harness for compatibility with Airtronics, KO, Kyosho, JR, Airtronics Z, and Hitec radios. *Refer to Figures 1 through 3 to convert plug.*

FIGURE 1 *With a small standard screwdriver, press on each of the three metal prongs until the wires are easy to remove. Remove wires.*

FIGURE 2 *With the screwdriver, carefully lift each of the metal locking tabs to the angle shown.*

FIGURE 3 *Insert each pin into the correct plug slot. Each pin should "click" into place. (Airtronics plug shown)*

The locking tab must not extend outside the plastic plug housing.

WHT – *White wire terminal (signal)*
BLK – *Black wire terminal (negative)*
RED – *Red wire terminal (positive)*
CAUTION *Improper installation of these wires may cause damage to the receiver, servo, and speed control.*

STEP 2
MOUNTING INSTRUCTIONS

1. DETERMINE BEST ESC MOUNTING LOCATION
The ESC should be positioned away from the receiver and antenna as shown in the Quick Set-Up photo above. Choose a mounting position that will keep the power wires as short as possible without obstructing movement of the suspension or the motor pod.
Remember, cooler operating temperatures mean higher efficiency. So, choose a mounting position that allows maximum airflow through the heat sinks.
2. INSTALL SPEED CONTROL
Use the included double-sided tape to mount the ESC.
3. INSTALL ON/OFF SWITCH
Determine a convenient place to mount the switch where it will be easy to get to. Mount the switch using a piece of double-sided tape or with a screw through the hole in the base of the switch housing.
4. INSTALL RECEIVER
Mount the receiver as far from the motor, power wires, battery, and servo as possible. These components all emit radio noise when the throttle is being applied. On graphite or aluminum, place the receiver on edge with the crystal and antenna as far above the chassis as possible. Mount the antenna close to the receiver and trim any excess wire off the top of the antenna.

STEP 3
SUPER ROOSTER HEAT SINK INSTALLATION

Heat sinks are required with the Rooster & Super Rooster for optimum performance and power handling. The Rooster heat sink comes factory installed and must not be removed. Included with the Super Rooster are heat sinks to fit onto the ESC's three separate transistor tab banks.

To Install Super Rooster Heat Sinks:

1. INSTALL THE HEAT SINKS. Place ESC on a flat surface and press the long/5-transistor heat sink onto the bank of 4 transistors on the upper left. Next, press the long/3-transistor heat sink onto the bank of 2 transistors on the upper right. The 2 remaining short/3-transistor heat sinks go onto the bottom bank of 6 transistors.
The heat sinks should press onto the transistors with a snug fit. If they are installed upside-down (longer fins up) or shifted off to one side, they will be too loose.
NOTE: Do not use too much force when installing the heat sinks because you can damage the transistors or other components on the PC board. Never use a vise or pliers to install the heat sinks.
2. DO NOT USE GLUE. Do not use glue or adhesives to attach the heat sinks to the transistors.
3. DO NOT SHORT CIRCUIT HEAT SINKS. The three banks of transistor tabs are separated by plastic on the case top. Each bank of heat sinks should never contact each other or other conductive objects (metal, etc.), or they will short circuit and damage the speed control.

NOVAK ELECTRONICS, INC.
18910 Teller Avenue
Irvine, CA 92612
www.teamnovak.com

53

STEP 4
HOOK-UP INSTRUCTIONS

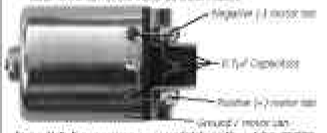
Refer to Quick Set-Up photos on front

1. INSTALL MOTOR CAPACITORS

   Electric motors generate radio noise that can interfere with your receiver and cause radio problems. Included in the ESC accessory kit are three 0.1μF (50V) non-polarized, ceramic capacitors. These capacitors must be installed on every motor to help reduce the noise generated by the motor and to prevent ESC damage.

   Solder 0.1μF (50V) capacitors between:
   - POSITIVE (+) motor tab & NEGATIVE (-) motor tab.
   - POSITIVE (+) motor tab & GROUND tab*.
   - NEGATIVE (-) motor tab & GROUND tab*.

   *If your motor does not have a ground tab, solder the capacitor leads to the can of the motor as shown below.



Extra 0.1μF capacitors are available in Novak kit #5620

2. IMPORTANT NOTE ABOUT SCHOTTKY DIODES

   NO SCHOTTKY DIODES

   Schottky diodes must NOT be used with reversible speed controls. Using a Schottky diode will damage the speed control and will void the warranty.

3. CONNECT SPEED CONTROL TO THE RECEIVER

   After the proper input plug plastic has been installed to match the receiver (Refer to Step 1), plug the speed control into the THROTTLE CHANNEL of the receiver.

4. CONNECT SPEED CONTROL TO THE BATTERY PACK

   ROOSTER—Plug the BT/Tamiya connector from speed control into a 6 or 7 cell battery pack (1.2 volts DC/cell). *Removal of BT/Tamiya connector voids warranty.

   SUPER ROOSTER—Cut the BLACK wire of ESC to the desired length and strip about 1/4" of insulation off the end. Solder to the *negative* side of a completely charged 6 to 10 cell battery pack (1.2 volts DC/cell). Cut the RED wire of ESC to the desired length and strip about 1/4" of insulation off the end. Solder to the *positive* side of the battery pack.

5. CONNECT SPEED CONTROL TO THE MOTOR

   ROOSTER—Plug the bullet connector on the YELLOW wire of speed control to motor *positive*. Plug the bullet connector on the BLUE wire of speed control to motor *negative*.

   SUPER ROOSTER—Cut the BLUE wire of ESC to the desired length and strip about 1/4" of insulation off the end. Solder to the *negative* tab of the motor. Cut the YELLOW wire of speed control to the desired length and strip about 1/4" of insulation off the end. Solder to the *positive* tab of the motor.

TIP: Twist BLUE & YELLOW motor wires once or twice as they go to the motor to reduce any radio noise emitted from power wires.

6. USING PLUGS FOR BATTERY & MOTOR CONNECTION

   High-quality low-resistance connector plugs, such as Dean's Ultra Plugs, can also be used to connect the Super Rooster. While these connectors make component changes quick and easy, the connection will never have the low resistance of a good solder joint.

   Use connectors that can not be connected backwards, as this will damage the ESC and void the warranty.

   It is good practice to use a female connector on battery to keep from short-circuiting on conductive surfaces.

   If you use connector plugs for battery and motor, use a male connector on the ESC battery wires and a female connector on the motor wires. Doing this will avoid plugging the battery into the motor output of the ESC.

STEP 5
TRANSMITTER ADJUSTMENTS

For proper speed control operation and programming, set transmitter adjustments as follows:

1. Set HIGH ATV or EPA to maximum setting.
   [Amount of throw at full throttle]
2. Set LOW ATV, EPA, or ATL to maximum setting.
   [Amount of throw at full brakes]
3. Set EXPONENTIAL to zero or middle setting.
   [Throttle channel linearity]
4. Set THROTTLE CHANNEL TRIM to middle setting.
   [Adjusts neutral position/increases or decreases coast brakes]
5. Set THROTTLE CHANNEL REVERSING SWITCH to either position.
   [Do not change switch position after programming]
6. Set ELECTRONIC TRIGGER THROW ADJUSTMENT to 50% throttle and 50% brake throw (or 5:5).
   [Adjusts pistol-grip transmitter's throttle trigger throw on electronic/digital transmitters]
7. Set MECHANICAL TRIGGER THROW ADJUSTMENT to position with 1/2 throttle and 1/2 brake throw.
   [Adjusts pistol-grip transmitter's throttle trigger throw on mechanical/analog transmitters]

STEP 6
SPEED CONTROL PROGRAMMING

Speed control should be connected to receiver and to a charged battery pack, and the transmitter adjusted.

1. TURN ON THE TRANSMITTER
2. TURN ON THE SPEED CONTROL
3. PRESS AND HOLD SPEED CONTROL'S SET BUTTON
   With transmitter throttle at neutral, press and hold the ESC SET button until the status LED turns *solid red*.
4. RELEASE ESC SET BUTTON WHEN LED IS RED
5. PULL TRANSMITTER THROTTLE TO FULL-ON POSITION
   Hold it there until the status LED *turns solid green*.
   NOTE: The motor will not run during programming even if it is connected to the speed control.
6. PUSH TRANSMITTER THROTTLE TO FULL-REVERSE
   Hold it there until the status LED *blinks green*.
7. RETURN TRANSMITTER THROTTLE TO NEUTRAL
   Status LED will *turn solid red*, indicating that throttle is at neutral and proper programming has been completed.

Speed control is programmed & ready to kick up a roost!

If transmitter settings are changed, it will be necessary to complete the programming sequence once again.

If you experience any problems during programming, turn off the speed control and repeat programming.

STEP 7
REVERSE DISABLE PROGRAMMING

Speed control should be connected to receiver and to a charged battery pack, and the transmitter adjusted.

1. TURN ON THE TRANSMITTER
2. TURN ON THE SPEED CONTROL
3. PRESS AND HOLD SPEED CONTROL'S SET BUTTON
   Press and hold the ESC SET button until the status LED *turns from solid red to solid green*.
4. RELEASE ESC SET BUTTON WHEN LED IS GREEN
5. PRESS SET BUTTON TO ENABLE/DISABLE REVERSE
   SLOW RED FLASH = REVERSE ENABLED
   FAST RED FLASH = REVERSE DISABLED
   Note: You must press the ESC SET button very soon after the LED begins flashing red (blink or two).
6. LED WILL TURN GREEN THEN EXIT PROGRAMMING
   Green LED indicates ESC is exiting programming mode.

TROUBLE-SHOOTING GUIDE

ESC Will Not Program Properly
- Too little transmitter throw—Increase ATV/EPA setting.
- Make sure ESC is plugged into the throttle channel of receiver. Check throttle channel operation with a servo.
- ESC SET button not held long enough—Press and hold SET button until status LED turns solid red.

ESC Will Not Go In Reverse
- Reverse circuitry disabled—Refer to Step 7 to enable.

Steering Channel Works But Motor Will Not Run
(Status LED is solid RED at all throttle positions)
- No signal from receiver—Make sure speed control is plugged into throttle channel of receiver. Check throttle channel operation with a servo. Check the wiring color sequence & metal socket insertion of receiver harness.

Steering Channel Works But Motor Will Not Run
(Status LED is RED at neutral / GREEN at full throttle)
- Check motor connections. Check motor and brushes.

Steering Channel Works But Motor Will Not Run
- Not programmed—Repeat programming.
- Thermal Shutdown—Allow to cool. Check for adequate airflow through heat sinks.
- Check wiring and connections—Check operation of system without speed control.

Receiver Glitches/Throttle Stutters During Acceleration
- Motor capacitors broken or missing—Refer to Step 4.
- Receiver or antenna too close to speed control, power wires, battery, or motor—Refer to Step 2.
- Bad connections—Check wiring and connectors.
- Graphite or Aluminum Chassis—Refer to Step 2.
- Excessive current to motor—Use a milder motor or a smaller pinion gear.

ESC Is Melted Or Burnt/ESC Runs With Switch Off
- Internal damage—Refer to Service Procedures.

For more help call our Customer Service Department.

SERVICE PROCEDURES

Before sending in your speed control for service, review the trouble-shooting guide and the instructions. The ESC may appear to have failed when other problems exist.

PLEASE NOTE: Speed controls that operate normally when received will be charged a minimum service fee and return shipping costs.

WHAT TO SEND: Fill out all information requested on the enclosed *REVERSIBLE ESC SERVICE CARD* (service card also available on our website) and return it with speed control.

WARRANTY WORK: For warranty work, you MUST CLAIM WARRANTY on the *REVERSIBLE ESC SERVICE CARD* and include a valid cash register receipt with purchase date on it, or an invoice from previous service work. If warranty provisions have been voided there will be a service charge.

SERVICE COSTS: Customer is responsible for all service costs (parts, labor, and shipping/handling charges). Speed controls will be returned by UPS/COD CASH ONLY. See *SERVICE CARD* for other payment and shipping options.

ADDITIONAL NOTES:
- Hobby dealers/distributors are not authorized to replace speed controls thought to be defective.
- If a hobby dealer sends your speed control for service, submit a completed *REVERSIBLE ESC SERVICE CARD* to the dealer and make sure it is sent with the speed control.
- Novak Electronics, Inc. does not make any electronic components (transistors, resistors, etc.) available for sale.
- To provide the most efficient service possible to our customers, it is not our policy to contact customers by phone or mail.

PRODUCT WARRANTY

The Rooster/Super Rooster is guaranteed to be free from defects in materials or workmanship for a period of 120 days from original date of purchase [illegible]

[illegible]

Because Novak Electronics, Inc. has no control over connection and use of the ESC, no liability may be assumed nor will be accepted for damage resulting from the use of this product. Every ESC is thoroughly tested and cycled before leaving our facility [illegible] By the act of connecting/operating ESC, the user accepts all resulting liability.

CUSTOMER SERVICE

CUSTOMER SERVICE HOURS (PST)

Monday-Thursday: 8:00am-5:00pm
Friday: 8:00am-4:00pm [illegible]

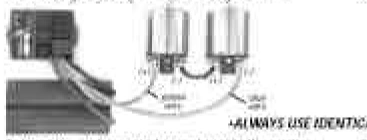
(949) 833-8873 • FAX (949) 833-1631

SUPER ROOSTER: DUAL-MOTOR SET-UPS & RECOMMENDATIONS

***** Using Dual-Motors is Not recommended with the Rooster *****

SERIES DUAL-MOTOR WIRING

This method gives you higher torque and longer run times.



•ALWAYS USE IDENTICAL MOTORS•

1. YELLOW wire from speed control goes to the positive tab on the first motor.
2. An extra piece of 14 gauge speed control wire then goes from the negative tab on the first motor to the positive tab on the second motor.
3. BLUE wire from the speed control goes to the negative tab on the second motor.

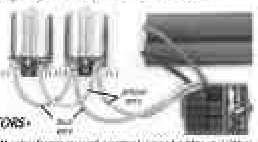
SERIES DUAL-MOTOR RECOMMENDATIONS

Special motors are available that are designed to be run together. These motors will give optimum performance and will also minimize the wear on your drive train.

Motors in series put the same load on the speed control as the total number of turns in both motors. For this set-up you can use motors with as few as 8 turns and the speed control will think its driving a 16 turn motor. You will get the speed and run times of a 16 turn motor with twice as much torque that is needed for heavier models.

PARALLEL DUAL-MOTOR WIRING

This method gives you faster speeds but shorter run times.



1. YELLOW wire from speed control goes to the positive tab on the first motor.
2. YELLOW wire then goes from the positive tab on the first motor to the positive tab on the second motor.
3. BLUE wire from the speed control goes to the negative tab on the first motor.
4. BLUE wire then goes from the negative tab on the first motor to the negative tab on the second motor.

PARALLEL DUAL-MOTOR RECOMMENDATIONS

As with the series set up, the motors that are designed for running in dual configurations will give you optimum performance and will minimize drive train wear.

Motors in parallel double the load on the speed control. For this reason you should not run motors with fewer turns than are available in a single motor. For example, if the lowest number of turns you can get is an 8 turn motor, each of your dual motors should be 15 or 16 turns.

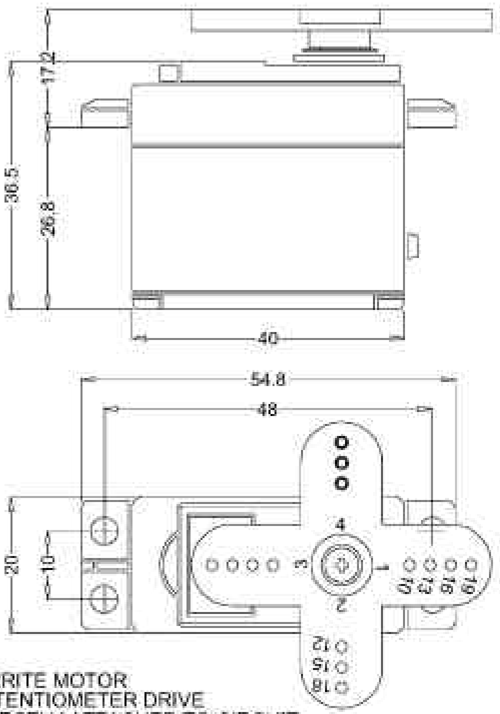
Appendix E:
Specifications for Hitec HS-303 Servo

Source: www.hitecrcd.com/support/manuals/servomanual.pdf

ANNOUNCED SPECIFICATION OF HS-303 STANDARD SPORT SERVO

1.TECHNICAL VALUES

CONTROL SYSTEM	: +PULSE WIDTH CONTROL 1500usec NEUTRAL	
OPERATING VOLTAGE RANGE	: 4.8V TO 6.0V	
OPERATING TEMPERATURE RANGE	: -20 TO +60°C	
TEST VOLTAGE	: AT 4.8V	AT 6.0V
OPERATING SPEED	: 0.19sec/60° AT NO LOAD	0.15sec/60° AT NO LOAD
STALL TORQUE	: 3kg.cm(41.66oz.in)	3.7kg.cm(51.38oz.in)
OPERATING ANGLE	: 45°/ONE SIDE PULSE TRAVELING 400usec	
DIRECTION	: CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec	
CURRENT DRAIN	: 8mA/IDLE AND 150mA/NO LOAD RUNNING	
DEAD BAND WIDTH	: 8usec	
CONNECTOR WIRE LENGTH	: 300mm(11.81in)	
DIMENSIONS	: 40x20x36.5mm(1.57x0.78x1.43in)	
WEIGHT	: 48.5g(1.71oz)	



2.FEATURES
- 3-POLE FERRITE MOTOR
- DIRECT POTENTIOMETER DRIVE
- MOTOR DIRECTLY ATTACHED TO CIRCUIT
- TOP RESIN BUSHING

3.APPLICATIONS
- SPORT/TRAINER AIRCRAFT
- CARS
- TRUCKS
- BOATS