DEVELOPMENT OF A NEURAL NETWORK BASED WEED SPRAYING SYSTEM

By

RONALD EDWARD EVANS

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1992

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1995

OKLAHOMA STATE UNIVERSITY

DEVELOPMENT OF A NEURAL NETWORK BASED

WEED SPRAYING SYSTEM

Thesis Approved:

Thesis Adviser

Dean of the Graduate College

ACKNOWLEDGMENTS

I sincerely wish to thank everyone who took part in this study, as well as everyone who helped provide me with this valuable educational opportunity. My appreciation goes to Dr. Marvin Stone and the Department of Biosystems and Agricultural Engineering for their support of this research. I also thank Mike Veldman for his generous help and words of wisdom, as well as Wayne Kiner, Gordon Couger, and everyone else at the lab.

Additional thanks go to my adviser Dr. Carl Latino for his guidance during this project and the writing of this document. Thanks also to Dr. R. Ramakumar for his help on my committee, to my mother Sharon for proofreading assistance, and to my wife Cathie for her support during this work. I also sincerely wish to thank Dr. Martin Hagan for his role in providing me with this important opportunity to gain practical engineering experience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

This thesis describes the design of a system that can distinguish between bare soil and live vegetation, and dispense a fluid onto the vegetation from a moving vehicle. Such a system would be useful to growers for detecting and applying herbicide to weeds in a field. During times of the year when there are no crops in a field, growers must control weeds because they deplete the soil of moisture and nutrients. A typical weed, such as bindweed (Convolvulus arvensis), spreads very quickly and can easily grow out of control. The traditional method of controlling such weeds is to drive a vehicle with a spray boom attached across a field and spray the entire field with herbicide. The spray boom used for this operation has several spray nozzles along its length which continuously dispense the chemical. If weeds cover only a small fraction of the field, then spraying the entire field is very wasteful of the expensive herbicide. In this situation, a system that would selectively spray only the weeds and not all the soil would be very useful. This is also a much more environmentally-sound solution, since much less of the herbicide would be released into the soil and air. Such a system requires a method of detecting the location of the weeds.

The objective of the system presented in this thesis is to provide a solution to the weed detection problem, and to implement the solution in a working sprayer system. As with the traditional spraying method, sprayers attached to a vehicle pass over the field. Here, however, each spray nozzle is controlled (turned on and off) by a computer that obtains information from a sensor.

As a sprayer unit passes over the field, the computer repeatedly receives information from a sensor that measures the spectral

characteristics of energy reflected from the ground surface. The sensor is sensitive to reflected energy in three wavelength bands. These bands are located roughly in the red, green, and near infrared regions of the electromagnetic spectrum. Using this spectral information, the computer program determines whether or not a plant is present within the viewing area on the ground under the sensor and turns a spray nozzle on or off accordingly in order to spray any detected vegetation.

Several methods to detect green vegetation based on spectral measurements of reflected radiation have been used in the past. Several studies, as discussed in Chapter II, suggest that reflected radiation levels in the red and near infrared wavelength bands are useful in plant detection. Some of these studies utilize simple formulas, called vegetation indices, involving these two quantities to detect vegetation. Since, for the system developed here, we wish to include the green portion of the spectrum to provide additional information and possibly improve performance over other systems not utilizing it, the vegetation indices involving red and near infrared are not employed. Another reason that the indices are not used here is that the application of a neural network to the problem was desired, as discussed below.

The system described in this thesis applies a mathematical construct called a neural network to the weed detection problem. In the application of a neural network, the system designer need not know the relationship between the input variables and the output variables. That is, we do not need to know the specific way in which the red, green, and near infrared reflected radiation levels vary with plant cover to apply a neural network to the problem. The neural network discovers the relationship during its training and the designer need only be concerned with the inputs and the outputs of the network.

Another reason for using a neural network in this application is that this research establishes methods that may be used in future work where more complex input data is analyzed. This future work may include developing ways to distinguish between weeds and regular crops.

A selective weed spraying system is useful in areas other than agriculture. For example, the system could be applied to controlling weeds on railways, roads, and airport surfaces.

CHAPTER II

PLANT DETECTION LITERATURE REVIEW

Several researchers have studied the detection of live vegetation using optical techniques. Summarized here are several of the papers on this work that have proven useful.

Wiegand and Richardson (1992) investigated the relationship between spectral observations and crop yields in agricultural landscapes. They found that the radiation reflected from live vegetation, in certain wavelength bands, contrasts significantly with that of surrounding materials such as soil and dead vegetation. The study indicates that such criteria might be useful in plant-detecting equipment.

Felton and McCloy (1992) describe a system called Detectspray, which is a reflectance-based system that controls the nozzles on a spray boom to apply pesticide or herbicide only to green vegetation. A sensor located in front of each nozzle views an area on the ground that is the same size as the spray pattern of the nozzle. The system relies on the fact that plants absorb red radiation and reflect well in near infrared. A detector pointing upward monitors incident solar radiation in these wavebands. Each sensor detects levels of red and near infrared reflectance, or radiation being reflected from the surface. A microprocessor at each sprayer unit calculates the fraction of radiation in each band that is reflected from the surface and calculates the ratio of near infrared reflectance to red reflectance. The computer compares this ratio to a threshold to decide whether a plant is present in the viewing area or not. Researchers claim that this system has been tested over a wide range of conditions with great success and that it can detect three percent green in the field of view. In other words, the claim is that the system can detect

vegetation when it covers three percent or more of the sensor viewing area. Also reported is that, in these tests, the savings in areas sprayed exceeded 90 percent. The paper indicates that the most important factor with the system is spray boom stability. This is a problem because of variation in height of the detectors over rough terrain. This "influences the size of the field of view and hence the size of weeds that are detected." Other problems noted are shadowing and lower reliability during the morning and late afternoon hours.

Shropshire, Von Bargen, and Mortensen (1990) conducted research on using an optical device for detecting weeds, which they call a reflectance ratio meter. The system uses photoresistors with optical filters, in an enclosure aimed at the ground, to detect irradiance in the red and near infrared bands. Electronic hardware produces a ratio of the near infrared illumination level to the red level. A range-interval method was devised to successfully identify weed presence from the reflectance ratio meter.

Nitsch, Von Bargen, Meyer, and Mortensen (1991) studied detection of weeds by a sensor for automatic control of a sprayer. This group examined reflectance curves in the 400-900 nm (nanometer) wavelength interval "to determine the contrast between weeds and other plants, soils and crop residues." Results showed that "the vegetative index, NIR/R, and the normalized difference rates, (NIR-R)/(NIR+R) at 800 or 850 nm provided the best method for velvetleaf detection." The study defines R as a measure of reflectance, or reflected radiation, in the red wavelength band. NIR is a measure of reflectance in the near infrared wavelength band. The two NIR bands that proved most useful, when used in the normalized difference rates calculation, were centered at 800 and 850 nm.

CHAPTER III

SYSTEM HARDWARE COMPONENTS

System Overview

The experimental sprayer system described in this thesis is intended to be driven across a field, and to detect and apply herbicide to weeds or any other green vegetation, but not to bare soil. The sprayer system consists of a tractor, a spray boom, five sprayer units, and a fluid system. The five sprayer units are mounted on the spray boom on the rear of the tractor, as shown in Figure 3.1.

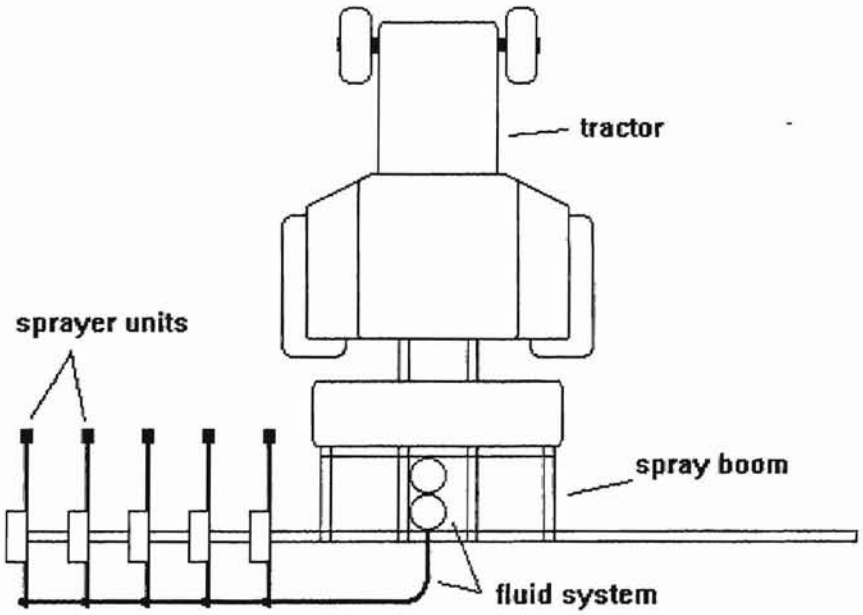


Figure 3.1: Tractor with spray boom.

Each of the five sprayer units consists of a sensor, computer, computer housing, spray nozzle, spray nozzle driver, and sprayer arm. Figure 3.2 shows one sprayer unit. The computer and spray nozzle driver are located inside the computer housing.

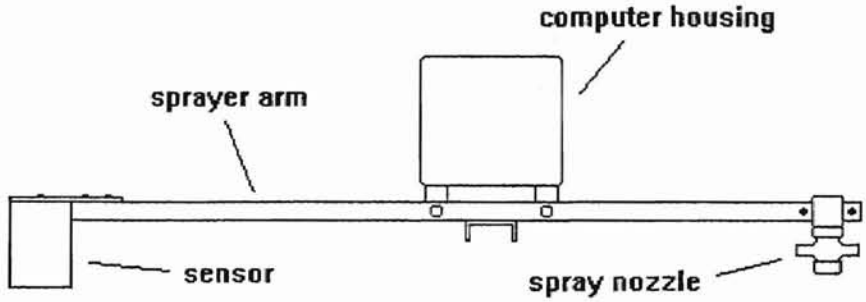


Figure 3.2: Sprayer unit.

Figure 3.3 shows a block diagram of a sprayer unit. As the weed sprayer system passes over a field, each sprayer unit independently detects and targets weeds on the ground.

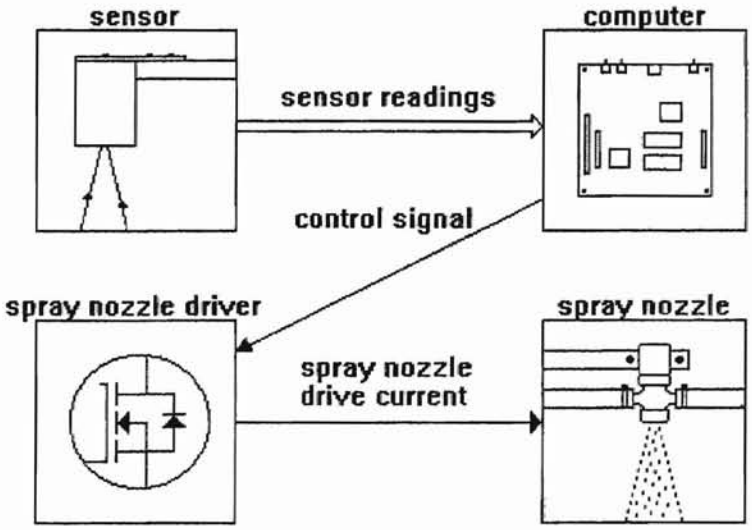


Figure 3.3: Sprayer unit block diagram.

To accomplish this, the sensor in each unit obtains optical data from the ground directly below it. The computer in the sprayer unit processes this data using a neural network based software algorithm. If a weed, or other plant, is detected, then the computer activates the spray nozzle at the appropriate time in order to spray the plant with herbicide.

Sensor

Each sprayer unit is equipped with a three-band optical sensor to obtain information from the ground surface. During operation of the sprayer system, the sensor passes over the ground and provides measurements of red, green, and near infrared radiation reflected from the surface. The computer reads this data for processing. Chapter IV provides more information on the sensor.

Computer

A separate microcomputer controls each sprayer unit. The computer is located inside the computer housing of the sprayer unit, as described in the Computer Housing section of this chapter. As the sprayer unit passes over the ground, the computer periodically acquires red, green, and near infrared reflected radiation readings from the sensor. The computer processes this data to determine whether or not a plant is present in the viewing area, and turns the spray nozzle on or off accordingly at the appropriate time. Chapter VII explains the algorithm in further detail.

The computer used for data processing in each sprayer unit is the Dearborn Network Analysis Tool (DNAT) manufactured by Dearborn Group, Inc. Appendix H provides information on the DNAT microcomputer in additional to that in this section. The computer is referred to in the appendix as the CNAT (Chrysler Network Analysis Tool), which is merely a different name for the same device. The DNAT single board microcomputer uses a Motorola MC68HC11A0 microprocessor as its control unit and has hardware that facilitates connection to a controller area network (CAN). The phase of the weed sprayer project described in this thesis does not utilize this CAN hardware. The computer board also provides headers with access to most of the pins on the MC68HC11A0, supports up to 64 kilobytes of memory, and has an RS-232 serial communications port. These three features are utilized in this project. The MC68HC11A0 contains eight 8-bit analog to digital converters, three of which are used to read the red, green, and near infrared reflected radiation measurements from the sensor. The processor also provides digital outputs that can be controlled from software, one of which is used to turn the spray nozzle on and off.

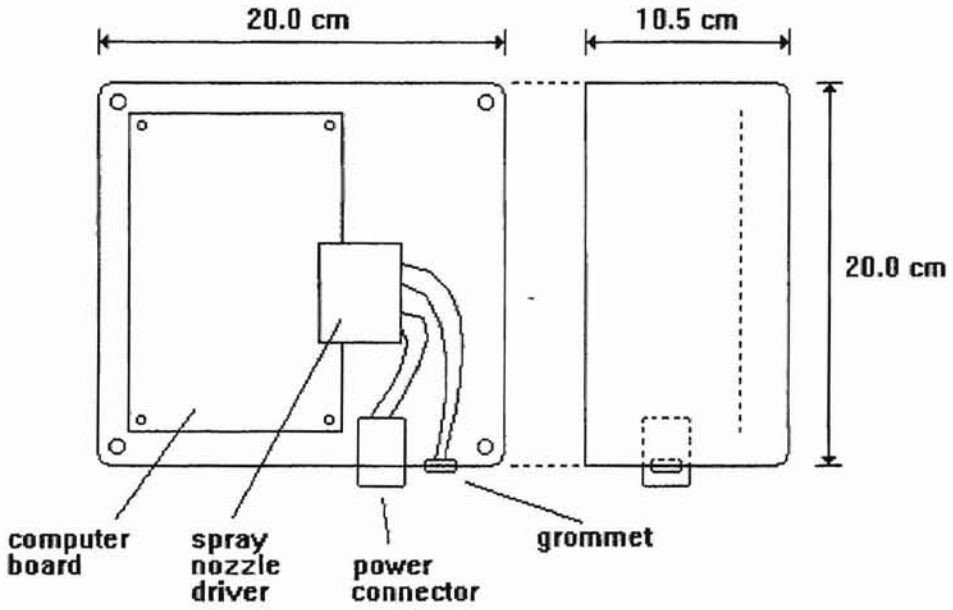
The DNAT computer can be programmed using two different methods. The first programming method is typically used during software

development and involves download of code by means of a serial (RS-232) communications link to a PC. The programmer compiles C code on the PC and sends the resulting Motorola s-record file to the DNAT board using a terminal program that supports text (ASCII) file transfers. The computer provides 32 kilobytes of non-volatile static RAM for storing such programs. The serial communications are controlled by the Buffalo monitor program residing in the 32 kilobytes of EEPROM (electrically erasable programmable read only memory), also provided on the board. The Buffalo monitor program was developed by Motorola to aid in developing systems based on the MC68HC11A0 microprocessor. The second method of programming the DNAT involves removing the EEPROM from the board, placing the user program on it with an EEPROM programmer, then placing the chip back in the DNAT board. This method is useful for creating stand alone programs that begin execution on power-up. This is the method utilized in the sprayer units for field operation.

Computer Housing

The computer housing is part of the sprayer unit, as shown in Figure 3.2. The housing is a plastic box that encloses the weed sprayer computer and spray nozzle driver to protect them from water, dirt, and other contaminants in an outdoor environment. Figure 3.4 shows the contents of the housing, with the housing cover removed.

Figure 3.4: Computer housing.



The housing is mounted to the weed sprayer arm with two bolts. The power supply wiring to the computer and spray nozzle driver pass through a connector in the bottom of the housing. The wiring from the spray nozzle driver to the spray nozzle, and from the computer to the sensor, passes through a grommetted hole in the bottom of the housing. The cover to the computer housing attaches with eight screws and forms a water-tight seal using a foam rubber gasket.

Spray Nozzle

The spray nozzle is part of the sprayer unit. It is an electrically controlled fluid valve which is located on the rear end of the sprayer arm as show in Figure 3.2. During weed sprayer operation, a rubber hose containing a pressurized fluid, such as a herbicide solution, is connected to the spray nozzle. Application of the proper electrical signal to the two terminals of the spray nozzle activates a solenoid, which opens a valve and dispenses the fluid toward the ground. The spray nozzle requires a 12 volts DC drive to activate, and draws about 0.7 amps current. The nozzle is closed when no voltage is applied. Appendix B shows the electrical connections to the spray nozzle.

Spray Nozzle Driver

The spray nozzle driver is a part of the sprayer unit and supplies power to the spray nozzle under control of a digital output from the weed sprayer computer. The spray nozzle driver is located on a circuit board that is attached to the computer board J2 header with a socket. The computer and spray nozzle driver are located in the computer housing. Figure 3.4 in the Computer Housing section of this chapter shows the location of the spray nozzle driver in the housing.

Appendix B shows a schematic diagram of the spray nozzle driver. An IRFD123 P-channel power MOSFET (metal oxide semiconductor field effect transistor) switches the power (12 volts DC at 0.7 amps) to the spray nozzle to turn it on and off. A 4N35 opto-isolator drives the gate of the MOSFET and is itself driven by the digital output PB0 from the MC68HC11A0 microprocessor in the weed sprayer computer. The opto-isolator is used to supply greater current to the MOSFET gate than is available at the computer's digital output, and to protect the computer from any high voltage transients that may occur as a result of switching the current on and off to the spray nozzle.

Sprayer Arm

The sprayer arm is part of the sprayer unit. The arm supports the sensor, computer housing, and spray nozzle, as shown in Figure 3.2. The arm of each sprayer unit is attached to the spray boom, as shown in Figure 3.1. The sprayer arm is 95.4 cm (37.6 in.) in length and is composed of 3/4 in. (1.9 cm) square steel tubing. The wiring from the computer housing to the sensor and spray nozzle pass through the sprayer arm. This serves to hide the wiring from view and to protect it.

Fluid System

The fluid system components of the experimental sprayer system are carried on the spray boom that supports the sprayer units. The fluid system consists of a pressurized air tank, a pressure regulator, a fluid tank, and rubber hoses. Figure 3.1 shows the location of these components, with the exception of the pressure regulator. The pressure regulator is located in-line between the pressurized air tank and the fluid tank. The pressure regulator serves to supply a constant pressure to the fluid tank. The fluid tank holds a herbicide solution, or water during testing, and supplies pressurized fluid to the sprayer units through rubber hoses. Each sprayer unit dispenses the fluid when it activates its spray nozzle.

The pressure regulator was set to supply a constant 25 pounds per square inch of pressure to the fluid system for most sprayer operations during the course of this project. The pressure at each spray nozzle affects the time required for the fluid to travel from the spray nozzle to the ground when the nozzle is activated. This affects the weed sprayer timing, and thus the ability of the unit to target plants detected on the ground.

Tractor

The tractor is used to carry the spray boom, to which the sprayer units are attached. The engine of the tractor is computer controlled and can be driven at nearly constant speed across a field in order to operate the weed sprayer system. The tractor provides a digital display of engine RPM, which allows the required speed to be repeated. Figure 3.1 shows how the spray boom and sprayer units are attached to the tractor.

Spray Boom

The spray boom is a metal structure that is attached to the rear of the tractor and supports the sprayer units. The spray boom also carries the fluid system. Figure 3.1 shows how the spray boom and sprayer units are attached to the tractor. The five sprayer units in the experimental system are placed at 18-inch intervals. This spacing is approximately equal to the width (perpendicular to the direction of travel) of the sensor viewing area and the width of the spray pattern of the spray nozzle. This spacing ensures that the entire ground surface under the sprayer units will be examined for the presence of weeds.

CHAPTER IV

SENSOR

Sensor Overview

A three-band optical sensor is the eye of a sprayer unit. During operation of the sprayer unit, the sensor passes over the ground and provides measurements of red, green, and near infrared radiation reflected from the ground surface. The computer reads this data for processing. Figure 4.1 shows the sensor mounted on a sprayer unit.

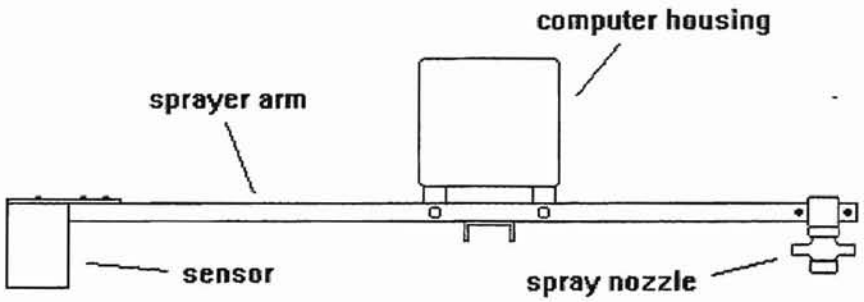


Figure 4.1: Sprayer unit.

Figure 4.2 shows the main sensor components and how they fit together. The main components are the sensor circuit board, circuit board mask, housing top plate, main housing, and aperture plate.

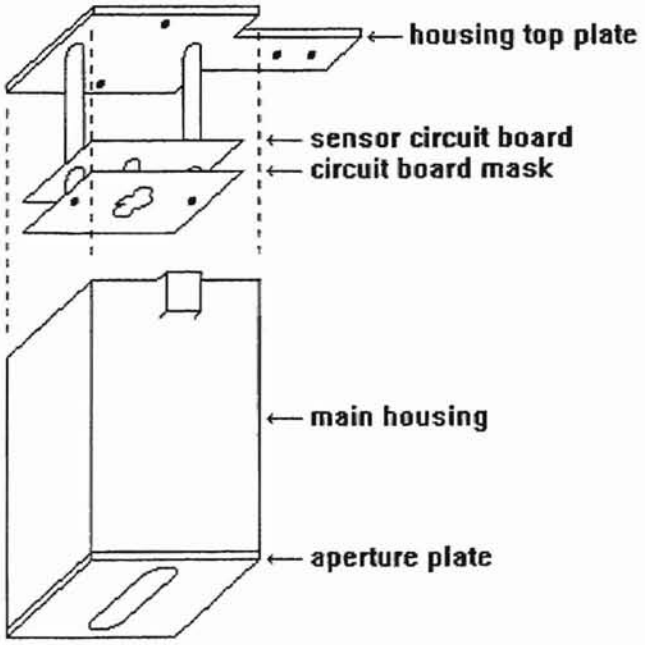


Figure 4.2: Main sensor components.

Basis of Plant Detection

This project utilizes the fact that live vegetation can be distinguished from bare soil by examining the intensity of electromagnetic radiation reflected from the ground surface at different wavelengths. Figure 4.3 shows a plot of relative intensity of reflected radiation vs. wavelength for a soybean plant and for soil. Other green plants show very similar reflectivity vs. wavelength curves (Nitsch 1991).

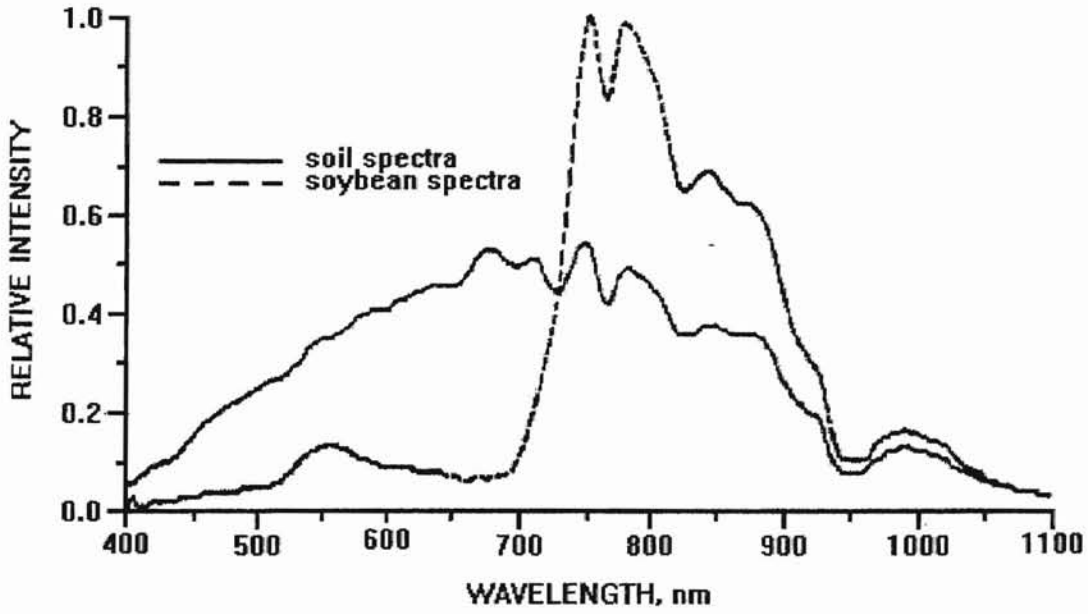


Figure 4.3: Reflected radiation spectra for soybean plant and soil. (Nitsch 1991)

The characteristic shape of the reflectivity curve for green plants is due primarily to the interaction of light with the chlorophyll in the leaves (Nitsch 1991). Chlorophyll tends to absorb light in the visible portion of the spectrum, specifically in wavelengths of about 400 to 700 nm. In contrast, it reflects very well in the near infrared, specifically from about 700 to 800 nm. The hump in the plant spectrum at about 550 nm shows that live vegetation

reflects the color green better than the other visible colors. The valley near 650 nm shows that vegetation absorbs red light very well. In contrast to live vegetation, the soil reflectance spectrum, also shown in Figure 4.3, has a more regular shape without the hump in green and the sharp transition between red and near infrared. These differences appear to provide a criterion for detecting plants against a background of soil. Several studies, as summarized in Chapter II, have indeed found that measuring reflected energy in the red and near infrared bands can be very useful in the detection of live vegetation. Use of the green band would appear to provide additional information.

Sensor Circuit

The sensor in each sprayer unit consists of three sections, each corresponding to one wavelength band. These three wavelength bands fall roughly into the portions of the electromagnetic spectrum that we call green, red, and near infrared. The three sections measure red, green, and near infrared radiation levels reflected from the ground surface. We may think of the sensor as seeing three colors, however the use of the term color here is somewhat inappropriate since one of the bands (near infrared) does not fall in the visible spectrum. The term "wavelength band" is used here instead of "color."

Each of the three sensor sections is sensitive to one wavelength band. Figure 4.4 shows the circuit diagram of one section of the sensor circuit. The other two sections are very similar. See Appendix B for a more detailed circuit diagram of the entire sensor circuit.

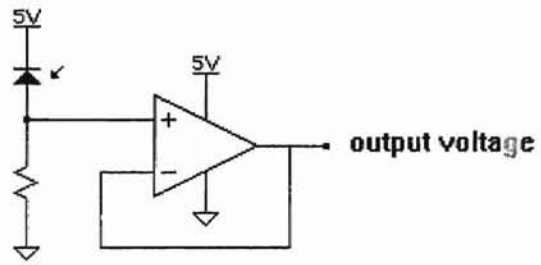


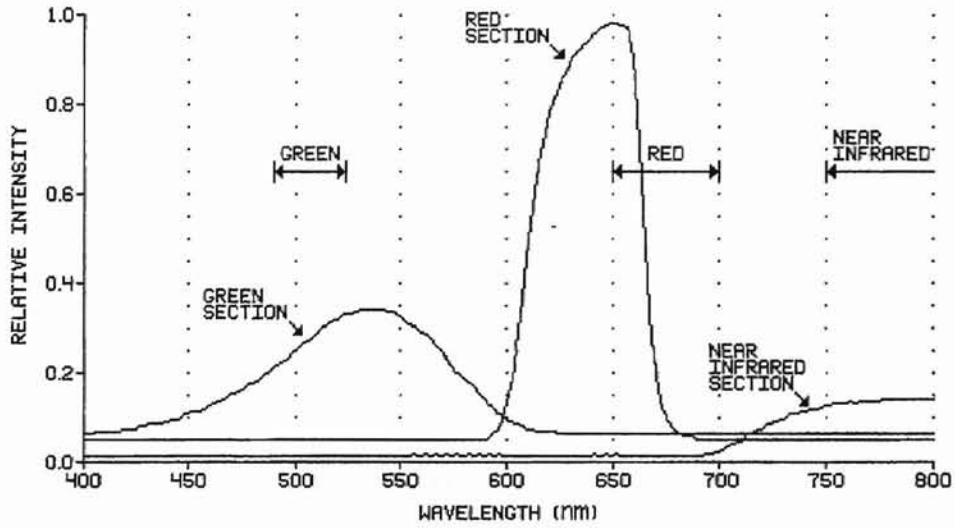
Figure 4.4: One section of the sensor circuit.

In each sensor section, a photodiode responds to radiation reflected from the surface of the ground. In the red section, a visible spectrum photodiode (Hamamatsu part no. G1115) with a colored glass filter (Spindler & Hoyer 370107) provides the measurement. In the red section, the same photodiode is used with a different colored glass filter (Spindler & Hoyer 370087). In the near infrared sensor section an unfiltered photodiode (Hamamatsu part no. S2164-01) is used

that is intended specifically to detect in the near infrared. No optical filter is necessary in the infrared section. Appendix F provides additional information on the photodiodes. Appendix G provides information on the filters used in this project, as well as on colored glass filters in general.

Figure 4.5 shows the spectral responses of the three sensor sections from a typical sensor.

Figure 4.5: Sensor spectral response.



Each hump in the plot corresponds to one section of the sensor. The three sensor sections correspond roughly to the portions of the electromagnetic spectrum that we call green, red, and near infrared. The green filter appears green to our eyes and the red filter appears red. Although no near infrared filter is used in the sensor, radiation in this band is invisible to the human eye. Also indicated on the plot are the regions of the spectrum that are generally called green, red, and near infrared.

The data for the sensor spectral response plot shown in Figure 4.5 was collected using special software installed in a sprayer unit. An instrument known as a grating monochromator provided an adjustable-wavelength light source that was directed at the sensor. The sprayer unit computer collected sensor readings as the monochromator was swept across the different wavelengths. The photodiode and filter data sheets in Appendix F and Appendix G provide additional information on the spectral responses of the individual parts.

Each photodiode in the sensor converts radiation reflected from the ground into an electric current (See the Theory of Photodiode Operation section of this chapter). Since, in the sensor circuit, each photodiode is in series with a resistor, the photodiode current causes a voltage across the resistor that is proportional to the radiation in the respective waveband.

In each sensor section, the voltage produced by the photodiode and resistor is buffered to the computer's analog to digital converter inputs through an LM324 operational amplifier stage with unity gain. The amplifier provides a low impedance output to the analog to digital converter, which reduces the possibility of electrical noise in the cable between the sensor circuit and the computer. Also, the amplifier provides a more stable high impedance input for the diode-resistor pair than would the analog to digital converter input alone.

The sensor circuit provides three output voltages, corresponding to red, green, and near infrared light levels reflected from the ground surface. These voltages may range from 0 volts to about 3.6 volts. This output swing is imposed by the LM324 operational amplifier, which is powered by a 5 volt single polarity supply.

Theory of Photodiode Operation

Each photodiode in the sensor transforms electromagnetic energy, in the wavelength band to which it is sensitive, to an electric current. The photodiodes are configured in reverse bias and the reverse saturation current in each photodiode is controlled by the intensity of energy that strikes the diode. The incoming photons create electron-hole pairs, which cause a reverse current through the diode. This current is proportional to the radiation intensity, in a certain wavelength band, falling on the device. Photodiode current, I_p, can be estimated using the following equation:

$$I_p = \eta q H,$$

where
η = quantum efficiency
q = charge on an electron, 1.6×10^{-19} C
$H = \phi A$ = radiation intensity (photons per second)
ϕ = photon flux density (photons per second per unit area)
A = photodiode junction area

Note that all factors on the right side of the equation above are constants except for the radiation intensity. Thus, the photodiode current is directly proportional to the radiation intensity, making the use of the photodiode a convenient way to measure light intensity.

Sensor Optical Geometry

Figure 4.6 shows the basic optical geometry of the sensor. For simplicity, the path of light for only one section of the sensor is shown. The sensor circuit, including the photodiodes, are mounted in an enclosure that allows light to enter in a particular way. A circuit board inside the sensor enclosure holds the three photodiodes so that their light sensitive sides face downward. A slot, or aperture, in the bottom of the enclosure allows light reflected from the ground to enter the box and strike the photodiodes. No other light is expected to enter the enclosure.

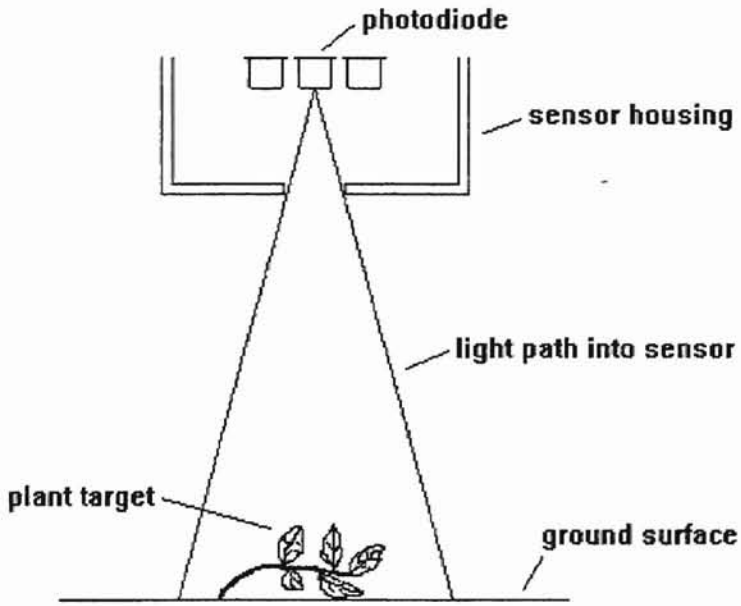


Figure 4.6: Basic sensor optical geometry. (dimensions not to scale)

The dimensions of the viewing area, or area on the ground that the sensor can see, is dependent on the optical geometry of the sensor. The size of the viewing area is affected by the distance of the photodiodes from the bottom of the aperture, the dimensions of the aperture, and the distance from the bottom of the aperture to the ground surface. These dimensions are assumed to be fixed for this

project. The first two dimensions are fixed by the physical construction of the sensor. The last is fixed by adjusting the height of the spray boom on the tractor so that the bottom of the sensor is the proper height above the ground. Although this height fluctuates as the tractor moves across a field, this change is assumed to not cause any major difficulties with the system. The width, or dimension perpendicular to the direction of travel, of the viewing area of each sensor section is about 22.5 inches. The length, or dimension in the direction of travel, is about 2.65 inches. Appendix A provides diagrams and calculations involving the sensor optical geometry.

Figure 4.7 shows the relationship between the viewing areas of the three sensor sections. Because the photodiodes in the three sensor sections lie next to one another on the sensor board, yet receive light from the same aperture, the viewing area of each is slightly different. Although this may affect sensor performance if vegetation is present near the ends of the viewing area, such an event is expected to occur so infrequently that the situation is acceptable. The viewing area of each sensor section is offset from that of the next section by about three inches.

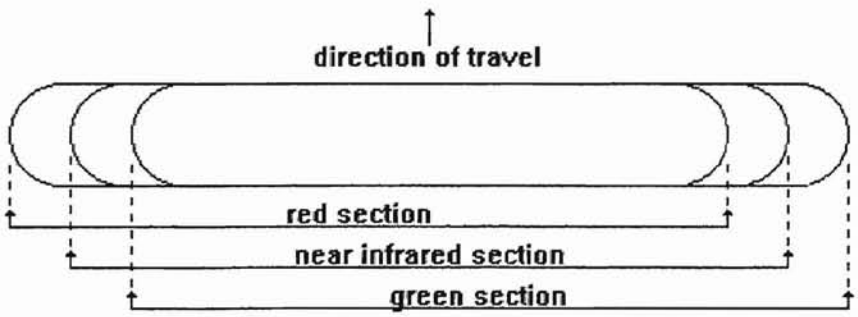


Figure 4.7: Viewing areas of the three sensor sections.

Each photodiode directly receives light, through the aperture, from a roughly rectangular area on the ground. The shape of the

viewing area actually has circular ends. This shape is due to the aperture in the bottom of the enclosure. Such a shape has no particular significance, but is simply easy to machine in the metal with a mill.

Each spray nozzle is intended to apply herbicide to the area on the ground that its sensor can see. The width (perpendicular to the direction of travel) of the viewing area of the sensor is approximately equal to the width of the spray pattern of the spray nozzle. The length (in the direction of travel) of the viewing area of the sensor is somewhat arbitrary. In general, the smaller the viewing area, the smaller the plant that can be detected. However, as the viewing area gets smaller, the operating frequency of the computer program must increase so that no portion of the ground surface is missed. Also, with a more narrow aperture, less light would strike the photodiodes and a higher voltage gain would be required in the sensor. This would tend to increase the effects of electrical noise in the sensor outputs and makes the design more difficult.

Sensor Circuit Board

The sensor circuit board provides a mounting place for all sensor circuit components. Figure 4.8 shows three views of the sensor circuit board. The bottom view shows the side of the board that faces the ground when the sensor is in service on a sprayer unit.

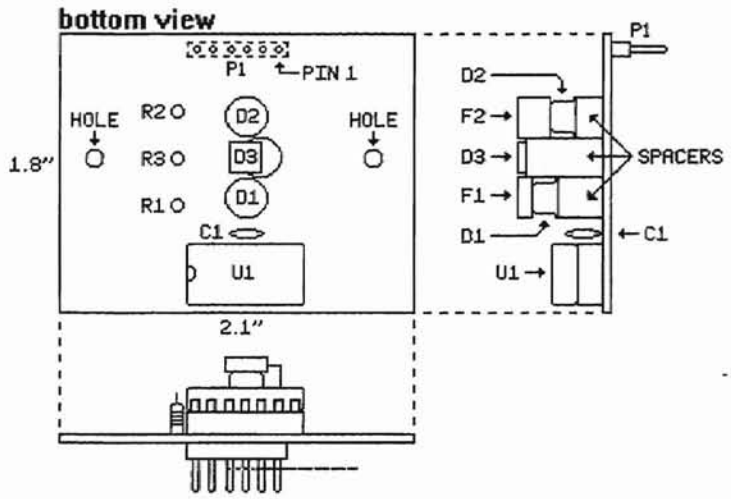


Figure 4.8: Sensor circuit board.

The three photodiodes D1, D2, and D3 in the sensor are glued to spacers and the spacers are glued to the circuit board. Plastic cement is used for these bonds. Soldering of the photodiode leads into the circuit board provides additional mechanical stability for the photodiode assemblies.

The red and green optical filters (F1 and F2) are bonded to the tops of the photodiodes (D1 and D2) with 5 minute epoxy at several points around the edges. No epoxy is near the center of the photodiode windows, where it would interfere with light reaching the active part of the device. The filters are cut from larger pieces to approximately

the size of the tops of the photodiodes. The top and bottom surfaces of these filters are the original polished surfaces. The near infrared photodiode D3, between the other two photodiodes, has no optical filter.

The plastic or fiber spacers, onto which the photodiodes are mounted, are cut to lengths so that the top surfaces of each of the photodiode assemblies are approximately in the same plane. This makes the distance from the ground to each assembly approximately the same so that the size of the viewing areas for the three photodiodes will be about the same. Hard plastic spacers were used to support the red and green photodiodes, and a fiber spacer was used to support the near infrared photodiode. These choices were based purely on the lengths of the spacers available.

The connector P1 on the sensor circuit board is a 5-pin single-row breakaway header with 0.1-inch centers. It allows connection of the sensor circuit to the computer board in the sprayer unit. Table 4.1 shows the P1 connector signal names and descriptions.

Table 4.1: P1 Connector Signals.

PIN	SIGNAL NAME	DESCRIPTION
1	+5V	Positive power supply to sensor circuit.
2	GND	Power supply ground to sensor circuit.
3	VRED	Output voltage from red sensor section.
4	VGRN	Output voltage from green sensor section.
5	VNIR	Output voltage from near infrared sensor section.

The LM324 operational amplifier integrated circuit U1 is a standard dual inline package (DIP) and is mounted in a standard socket. The board on which the components are mounted is a standard pad-per-hole epoxy glass composite prototyping board that is cut to size.

Sensor Circuit Board Mask

The sensor circuit board is partially covered with a mask to prevent light from striking any portion of the board except the photodiodes and a small area around them. Figure 4.9 shows the shape of the mask.

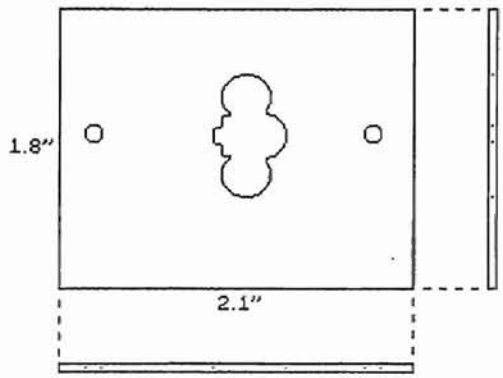


Figure 4.9: Sensor circuit board mask.

This device was deemed necessary after tests showed that light reflected from the metal on the circuit board affected sensor readings. The mask is the same size as the sensor circuit board and has an odd-shaped hole cut near the center to allow the photodiode assemblies to be exposed to light. The hole in the center is custom-shaped to fit around the photodiode assemblies and allow them to extend through the mask a small distance. The mask is made of standard copper-plated glass epoxy composite circuit board. This material was chosen because of its availability and because it is easy to cut. The copper plating provides no known benefit. The mask is painted flat black on both sides to minimize reflection of light. Plastic spacers hold the mask about 3/8 inches away from the sensor circuit board, allowing the photodiode assemblies to extend about 1/16 inches past the mask. These dimensions are not critical.

Sensor Housing

The sensor housing encloses the sensor circuit assembly. The housing is a metal box with three main pieces--the top plate, the main housing, and the aperture plate. Figure 4.2 shows how these three pieces fit together.

The top plate of the sensor housing, as shown in Figure 4.10, provides support for the sensor circuit assembly and secures the sensor to the sprayer unit arm.

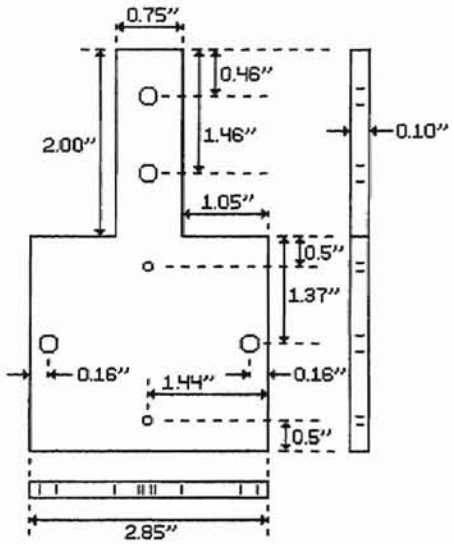


Figure 4.10: Sensor housing top plate.

This part is made of 14-gauge steel. It is fixed to the main housing with two screws, and to the sprayer unit arm with two screws. The sensor circuit assembly, with the mask attached, is suspended from the housing top plate by 1.5-inch spacers, as shown in Figure 4.2. The position of the sensor circuit board in the housing provides the proper optical geometry for the system. The Sensor Optical Geometry section of this chapter describes this spacing.

The main housing, as shown in Figure 4.11, is made of 2.5-inch square aluminum tubing. This part protects the sensor circuit

assembly (circuit board and mask) and provides a mounting place for the aperture plate. Two threaded holes at the top of the main housing allow attachment of the sensor housing top plate. Two small aluminum blocks welded to the main housing provide enough thickness for the two holes to be tapped. A rectangular cut at the top allows passage of the sensor cable into the end of the sprayer unit arm.

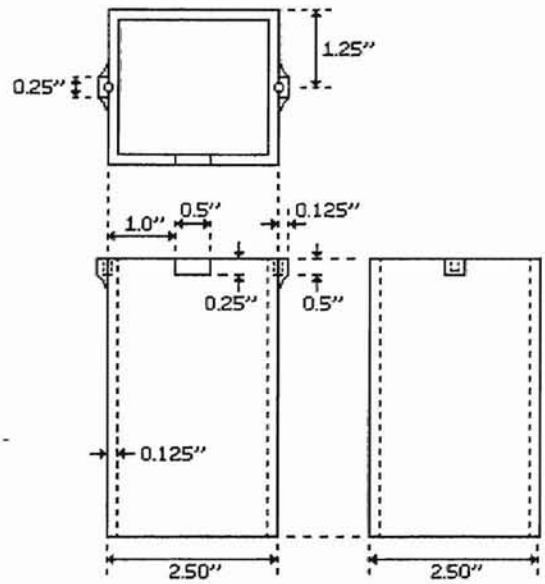


Figure 4.11: Sensor main housing.

Figure 4.12 shows the aperture plate. This part is a 2.5-inch square of 0.125-inch aluminum sheet, with a slot that allows light from only a certain area on the ground to strike the detectors in the sensor. The plate is welded onto the bottom end of the main housing. The outer edge of the aperture plate is beveled so that the part fits into the end of the housing. This bevel aids in correct placement of the plate. The Sensor Optical Geometry section of this chapter describes the geometry associated with the aperture.

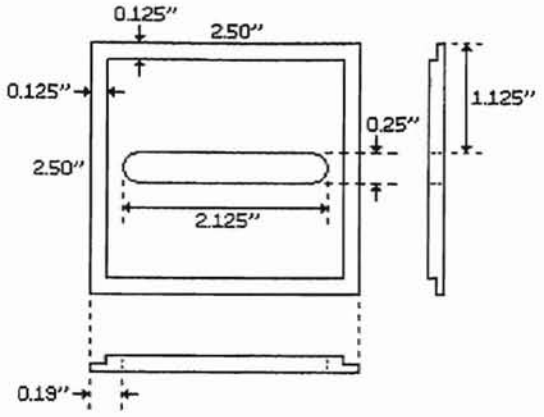


Figure 4.12: Sensor housing aperture plate.

CHAPTER V

NEURAL NETWORKS BACKGROUND

Introduction to Neural Networks

Neural network theory has developed as a result of man's desire to build intelligent machines. Since the only object that we generally view as intelligent is the human brain, simulation of this true neural network with an artificial one seems a good place to start. Neural networks are, in fact, an attempt to simulate the interaction of the nerve cells, or neurons, in the human brain. Although implementation of an artificial neural network as complex as the natural one found in the human brain is not currently practical, much simpler networks can learn some useful tasks that some consider intelligent.

Neural networks are generally mathematical constructs implemented in software, although some have been implemented in hardware as integrated circuits. Neural network development is inspired by known facts about the brain. They are characterized by having the following properties (Rich 1991):

1. A large number of very simple neuron-like processing elements, or nodes. The interaction of the many elements gives the network its power.
2. A large number of weighted connections between the elements. The weights on the connections encode the knowledge of the network.
3. Highly parallel, distributed control.
4. An emphasis on learning internal representations automatically.

Neural networks are theoretically capable of learning nearly any task imaginable. They learn by example. That is, a network is taught a particular task by being presented with many example problems and their corresponding solutions. By adjusting its internal structure, the network learns to solve the problems. A set of inputs to the network characterizes the problem and the network's output is its solution to the problem.

Neural networks are an emerging technology and have recently been applied to a wide variety of problems. They have successfully been used in bankruptcy prediction, real-time system control, speech recognition, machine vision, game playing, classifying sonar signals, and driving a vehicle.

Many different neural network structures exist. This chapter, as well as the neural network development work for this thesis, deals with fully connected, layered, feedforward networks. This is a very common neural network structure that has been successfully applied in many areas. Many different training methods are also available. The work for this thesis deals with a training method called back error propagation, or backpropagation. Many variations of backpropagation learning exist, some of which are mentioned in other parts of this thesis. This basic training method was chosen because of its success and popularity, and because of the availability of neural network development software utilizing this method.

The Neural Network Node

A neural network generally consists of several simple processing elements, or nodes. Each node performs a simple computation. It is the interaction of the entire group of nodes that gives the network its strength. Neural network nodes are modeled after neurons in the brain. Figure 5.1 shows a simplified diagram of a natural neuron. Figure 5.2 shows a neural network processing element, which is similar in structure to the natural neuron.

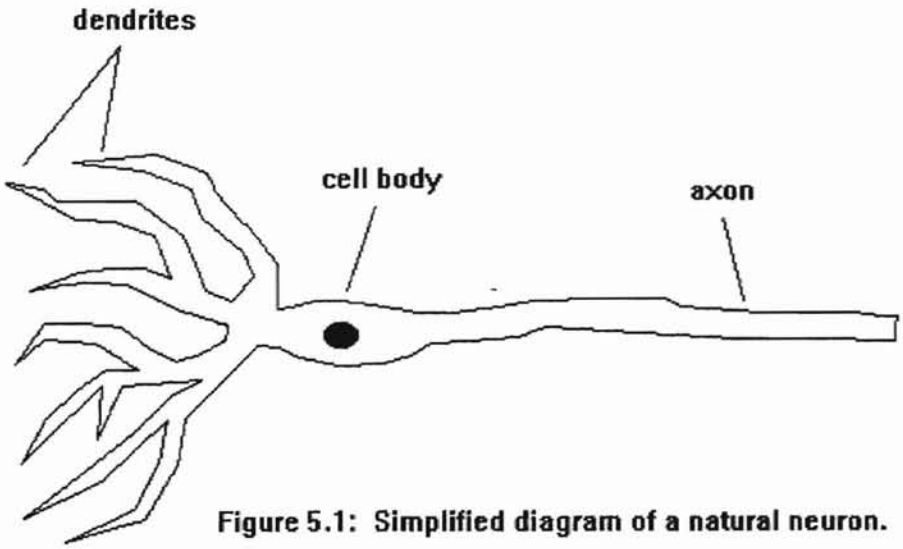


Figure 5.1: Simplified diagram of a natural neuron.

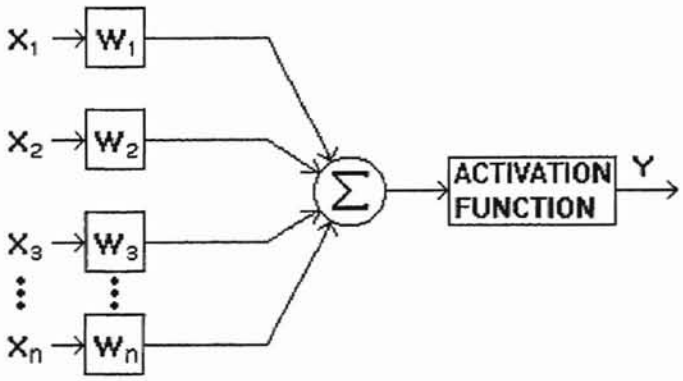


Figure 5.2: Neural network processing element, or node.

Each true node in a neural network sums its inputs and passes the sum through an activation function, or transfer function, to produce an

output. The inputs X_i to the node are on the left and the output Y is on the right. Each input X_i comes from the output of a node in the previous network layer, except the bias input, which is commonly set to unity. Every node has this bias input. Each input connection, including the bias input, is weighted by a real number W_i. The output Y of a node is given by the following expression:

$$Y = f(S),$$

where

f = activation function of the node.
S = weighted sum of the node inputs.
$= \sum_i (X_i * W_i)$

A common activation function for a neural network node is the sigmoid (or S-shaped function), which is defined by the following expression:

$$f(x) = 1 / (1 + e^{-x})$$

Figure 5.3 shows a plot of the sigmoid function. The activation function of a node is required by the backpropagation learning algorithm to be continuous and differentiable (Rich 1991). Other activation functions that are sometimes used are the hyperbolic tangent and the sinusoid.

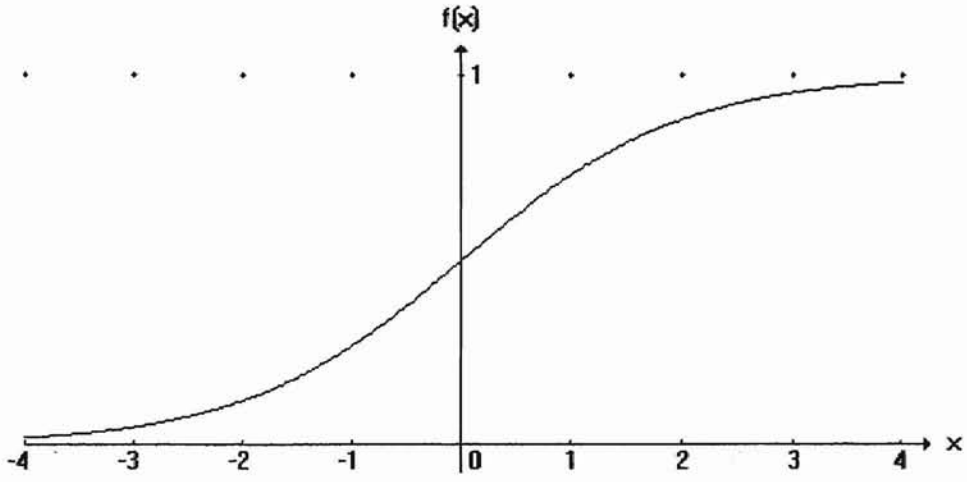


Figure 5.3: The sigmoid function.

Neural Network Structure

A neural network is composed of rows, or layers, of nodes. A layer may have any number of nodes. The inputs to each node are the outputs from all the nodes in the previous layer, in addition to the bias input.

Figure 5.4 shows the general structure of a neural network. The column on the left is called the input layer and receives external inputs. The units in the input layer are not true nodes and do not perform any computation. They merely pass the inputs into the network. All the units in the other layers are true nodes and function as described in the section of this chapter entitled The Neural Network Node.

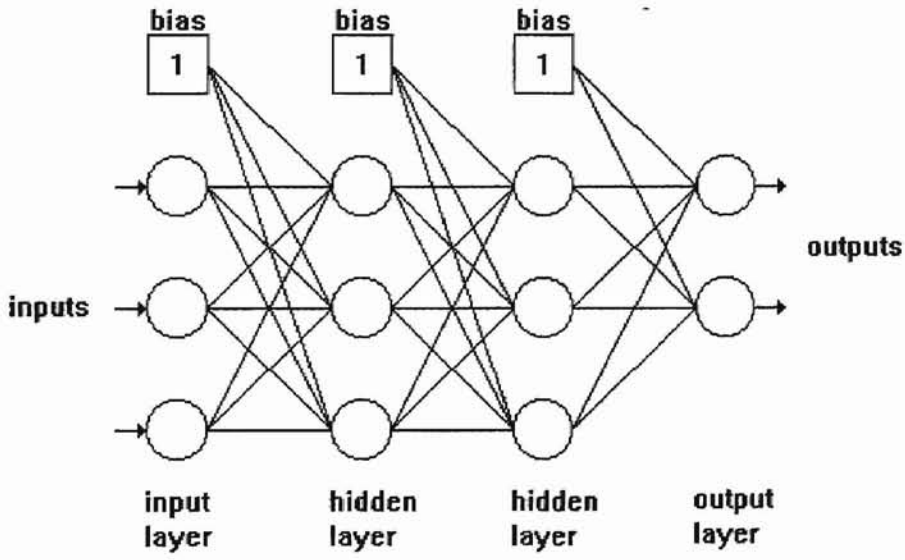


Figure 5.4: General structure of a neural network.

The layer on the right of the diagram is called the output layer. The nodes in the output layer provide the outputs of the network.

These numerical outputs would generally be evaluated by a software program to get information from the neural network.

The layers between the input and output layers are called the hidden layers. Neural networks generally have one, two, or three hidden layers. The number of layers required for a particular application, as well as the number of nodes required in each layer, is related to the complexity of the relationship between the inputs and the expected outputs. Many applications require only one hidden layer (*Neural Computing* 1991).

Neural Network Learning

In order for a neural network to be useful it must be trained to perform some task. Training consists of modifying the weights, or strengths of the connections, in the network. Training forms the mapping between the inputs and outputs that is required for the particular application. Typically the training process is carried out by a neural network development software package.

Neural network learning requires that the designer collect a training data set and a testing data set. These data sets consist of many sets of example inputs along with the corresponding expected outputs. This data provides sample problems and their corresponding solutions. This data should be representative of the types of problems the network would encounter in its final application. The neural network is trained using the training data set. Then performance of the trained network is evaluated using the testing data set. A neural network should not be evaluated using the same data on which it was trained because this may only show how well the network memorized the training set. The network should be able to generalize to the new situations in the testing set.

Training has a forward propagation step and a backward propagation step. During the forward propagation step a set of inputs from the training data set is passed through the network. The network produces outputs, which are compared to the expected outputs to produce a measurement of error. During the backward propagation step, the weights of the network are modified according to a certain learning rule in order to reduce this error. This cycle, consisting of a forward propagation step and a backward propagation step, is carried out for every input-output pair in the training data set. Generally the training set is passed through many times in this manner to properly train a neural network.

Testing of a neural network is carried out after training has stopped. The purpose of testing a trained neural network is to determine how well the network can perform the task for which it is intended. The weights remain fixed during testing. It consists of applying only the forward propagation step to each input-output pair in the testing data set. As in training, each set of inputs is passed through the network and the actual outputs are compared with the expected outputs. Here the testing data set is used instead of the training set. A calculation of error for the entire testing set, such as the mean squared error, may be useful in evaluating how close the neural network outputs are to the expected outputs. In a neural network that is to be used for classification, the designer may want to tabulate how many of the test cases the network correctly classified after it was trained. The level of performance required for the trained neural network, as well as the method of evaluating a trained neural network, will generally be different for each application. If testing shows that performance is unsatisfactory, additional design and training work may be required.

Using a Trained Neural Network

After a neural network has been trained and its performance with the testing data set is found to be adequate, the network can be placed in service to perform the task for which it was intended. This normally means implementing the neural network as a series of mathematical calculations in software.

External inputs are applied to the neural network in order to utilize it. This input data might be stock prices, statistics found during research, or measurements taken by a sensor, for example. The neural network processes the set of inputs and produces a result at each network output. These outputs may be used in different ways for different applications. For neural networks used in the classification of a set of inputs into certain categories, typically the network output that is numerically greater indicates the network's decision. For example, a neural network trained to recognize the written digits 0 through 9 might have 10 outputs, each corresponding to one of the digits. The output that is numerically greatest might indicate the digit recognized by the network, in response to an image scanned from a sheet of paper.

CHAPTER VI

SPRAYER UNIT NEURAL NETWORK

Neural Network Overview

The neural network in a sprayer unit is a mathematical construct that is part of the software. It is the neural network that ultimately decides, based on the red, green, and near infrared sensor readings, whether or not a plant is present in the viewing area of the sensor. The neural network consists of a set of numerical constants, called weights, and a series of equations that perform calculations using these weights and the three sensor readings. The results of the final calculations, at the outputs of the neural network, determine whether or not a plant has been detected.

The neural network development for this project concentrated on neural networks that are considered fully connected, layered, feedforward networks that are trained using a method called back error propagation, or backpropagation. This neural network structure and training method is popular and has found many successful uses.

Neural network development consists of building, training, and testing. Building creates an untrained neural network that is, in general, capable of no useful task. During training, an attempt is made to "teach" the neural network to learn a particular task. The task required for this project is plant detection. Testing the trained neural network determines how well it has learned to perform its required task. Neural network building, training, and testing were performed for this project with the help of neural network development software called Neural Works Professional II/Plus, from a company called Neural Ware. This is a very extensive neural network development tool that allows great flexibility in network configuration and training.

Because of the unique nature of this application of neural networks, determining a good network configuration and training procedure for the task of plant recognition consisted largely of trial and error. Tests were run using many variations of configuration and training. The development software literature (*Neural Computing* 1991) gave some direction on parameters that normally work well. A building, training, and testing cycle consists of building a neural network with a particular configuration, training it with a certain number of training examples from the training data set, then testing it on the test data set. These data sets consist of many examples of typical input-output pairs for the neural network.

See Chapter V for more information about neural networks in general. See the Basic Neural Network Implementation section in Chapter VII for details on the operation of the final neural network used in the sprayer unit.

Neural Network Building

This project deals only with the class of neural networks that are fully connected, layered, feedforward networks. This is a popular class of neural networks that has been used successfully in a wide range of applications. Chapter V provides more information on this class of neural networks. To design, or build such a neural network, the user must first specify the number of hidden layers and the number of processing elements, or nodes, in each layer. The number of nodes in the input and output layers is essentially fixed by the neural network application. The number of hidden layers (the layers between the input and output layers) to use and the number of nodes to use in each hidden layer is not so clear.

The number of nodes in the input layer of the neural network is the same as the number of inputs to the network. Since the sensor provides three reflected radiation readings (red, green, and near infrared), there are three inputs to the neural network and three nodes required in the input layer. Typical neural networks used for classification have one output node for each possible classification (*Neural Computing* 1991). For this application, the network must decide either "no plant" or "plant," so there are two output classifications, and thus two nodes in the output layer. One output corresponds to the "no plant" condition and one to the "plant" condition. The network's decision is indicated by the output that is numerically greater. For example, for a given set of red, green, and near infrared sensor readings, if the "plant" output is greater, then the neural network has decided that a plant is present in the viewing area. Many variations were tried in the number of hidden layers and the number of nodes in each hidden layer.

The neural network designer must also specify the transfer function, or activation function, of each node, the learning rule (how the weights are modified during training), and how many total training examples will be presented during learning. Also, the user may specify how many training examples are presented to the network between weight updates. The development software calls this figure the epoch length. The development software used for this project allows specification of many other parameters for neural network building and training, but in the application these parameters were left at their default settings for simplicity. The Neural Network Development Results section of this chapter provides more information on neural network building for this project.

Neural Network Training

After a neural network configuration has been set up, or built, the development software trains the network. The training modifies the weights, or strengths of the connections, within the network. Before training, the weights are set to small random numbers. Then, during training, input-output pairs from the training data set are repeatedly passed through the network and the weights are modified so that the output error is reduced. The general learning method for all neural network training for this project is called backpropagation. The backpropagation method requires a specific learning rule. The development software provides many learning rules to choose from, all of which were tested for applicability to this project. A learning rule was chosen at the beginning of the training of each network configuration, and was not changed for the duration of the training of that neural network. Study of the mathematics behind backpropagation and the associated learning rules used by the development software for neural network training was not necessary for this project. The learning rules are referred to here by name only. The development software literature (*Neural Computing* 1991) describes the backpropagation method for each learning rule in further detail. Rich and Knight (1991) also provide information on backpropagation. The Neural Network Development Results section of this chapter provides more information on neural network training for this project. See the Neural Network Training and Testing Data section of this chapter for more information on this data.

Neural Network Testing

After a neural network has been trained in a certain way using the training data set, its performance may be tested using the test data set. A network is not exposed to the test data during training. The network's performance on the test data provides a measure of the network's ability to generalize to new situations, as it will have to do in service in a sprayer unit. The neural network development software carries out the testing of a trained network by presenting it with the test data and recording the network outputs for each test case. This data can then be analyzed further to determine how many of the test cases the neural network correctly classified. The Neural Network Development Results section of this chapter describes the tests that were run for this project. See the Neural Network Training and Testing Data section of this chapter for more information on these two data sets.

Neural Network Development Results

Nearly all of the neural network development for this project was carried out using training and testing data from a sensor that was an early prototype of the sensor described in this thesis. The earlier sensor, referred to as Prototype 1, used the same circuit, but a different enclosure, than the later Prototype 3, and did not include the sensor circuit board mask for limiting reflections inside the sensor housing. The newer metal enclosure is more easily manufactured and is stronger than the earlier plastic enclosure. Very little additional neural network development was required once training and testing data was recorded for the new sensor prototype (Prototype 3) described in this thesis.

As described in the previous sections, neural network development consists of iterative building, training, and testing using the development software. Each cycle through building, training, and testing produces a trained neural network and an evaluation of that network's ability to perform the task of plant recognition required for a sprayer unit. Building and training the neural networks that were tested for this project required specification of the following six main parameters:

1. Number of hidden layers.
2. Number of nodes in each hidden layer.
3. Transfer function (activation function) for each node.
4. Epoch length (number of training examples presented during training between weight updates).
5. Learning rule used during training.
6. Number of training examples presented during training.

The development software allows specification of many other parameters for a neural network, but these were left at their default settings for simplicity of testing. The number of input nodes (three) and output nodes (two) is fixed by the application, as described in the Neural Network Building section of this chapter. The neural network

development for this project consisted primarily of trial and error using variations of the six main parameters, as listed above, during building and training. Each resulting neural network was then tested using the sample cases in the testing data set. The fraction of the sample problems in the testing data set correctly classified by the network gives a measure of the trained neural network's performance.

Approximately 440 building/training configurations were tried in an attempt to find the best neural network for use with the Prototype 1 sensor. Following is a summary of the different variations that were tried for each neural network building and training parameter:

Number of hidden layers: 1 to 3.
Number of nodes in each hidden layer: 1 to 10.
Transfer function of each node: linear, sigmoid, sine, hyperbolic tangent.
Epoch length: 1 to 80.
Learning rule: delta, cumulative-delta, normalized-cumulative-delta.
Number of training examples: 1000 to 350000.

The best trial using Prototype 1 resulted in a neural network that correctly classified 88.3% (53/60) of the test cases in the testing data set. This neural network has the following characteristics:

Number of hidden layers:	2
Number of nodes in each hidden layer:	3
Transfer function for each node:	hyperbolic tangent (tanh)
Epoch length:	1
Learning rule:	normalized-cumulative-delta
Number of training examples presented:	40000

The Prototype 3 sensor is a result of improvements, as mentioned above, to the Prototype 1 sensor. The training and testing for Prototype 3 used different training and testing data sets, which are the data sets described in the Neural Network Training and Testing Data section of this chapter. Using exactly the same parameters for neural network building and training with this new data resulted in a network that correctly classified 100% (80/80) of the test cases in the new testing data set. By changing the epoch length from 1 to 16, a neural network

was developed that was deemed slightly better because the actual network outputs in the test results file appeared, in general, somewhat closer to ideal. The actual improvement here was not quantified by any concrete means. The final trained neural network used in the sprayer unit described in this thesis was built and trained using the following main parameters:

Number of hidden layers:	2
Number of nodes in each hidden layer:	3
Transfer function for each node:	hyperbolic tangent (tanh)
Epoch length:	16
Learning rule:	normalized-cumulative-delta
Number of training examples presented:	40000

The weed sprayer neural network is considered a layered, fully connected, feedforward network. Figure 6.1 shows a diagram of the network. The Basic Neural Network Implementation section of Chapter VII provides more information on the final weed sprayer neural network.

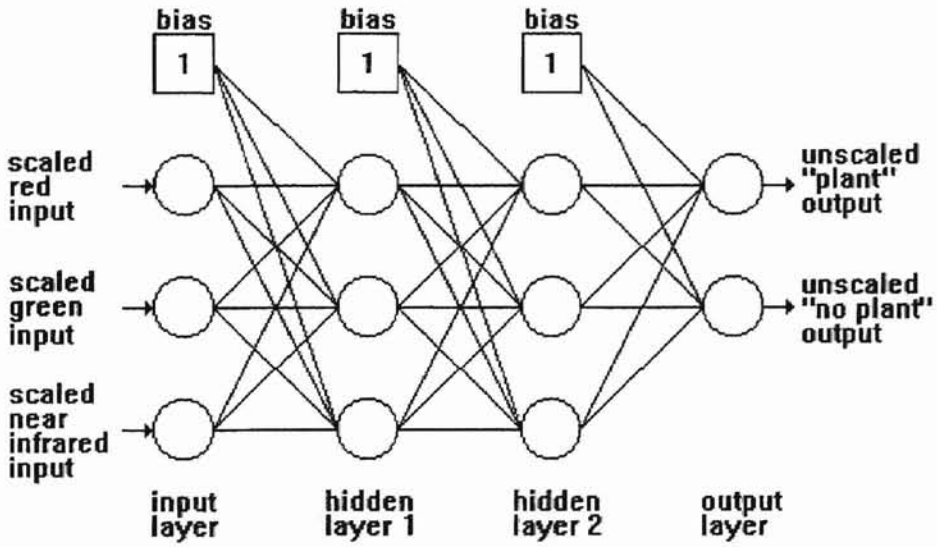


Figure 6.1: Sprayer unit neural network.

Neural Network Training and Testing Data

Neural network training and testing requires training and testing data sets. Each of these data sets consists of a group of input-output pairs that are typical of those that the neural network might encounter in service. After the network has been trained, it must be tested with input-output pairs that it has not encountered before. This testing measures how well the network can generalize to new situations. The training and testing requires the collection of many typical red, green, and near infrared reflected radiation sensor readings with a plant present and with no plant present, under diverse conditions. The conditions varied for the collection of this data were soil type, lighting, soil moisture, and plant density.

In order to obtain typical sensor readings for the training and testing data sets, field conditions were simulated using a test fixture. During data acquisition, a structure suspended a weed sprayer sensor above a platform where soil and plants were placed. The soil spread on the platform simulated the ground in a field. Uprooted bindweed (Convolvulus arvensis) plants placed on the soil under the sensor simulated live bindweed. The data was collected from the sensor using an ACRO data acquisition computer and a PC. The ACRO has 17-bit analog-to-digital converters to collect such data. The raw data was later converted to 8-bit precision for use with the neural network. This was done because the MC68HC11A0 analog-to-digital converters that acquire data in the sprayer unit have 8-bit precision. The neural network should be trained and tested on data similar to that which it will encounter in service.

Appendix E lists the training and testing data sets collected with the test fixture. Each data set consists of 80 sets of input-output pairs representing example inputs and outputs of the neural network. Following is an example line from one of the data sets:

```
120   99    126   1     0      ! 1has0201
```

Each line of data consists of a red, green, and near infrared sensor reading, followed by the two idealized expected outputs of the neural network. If a plant were in the viewing area of the sensor when the data was recorded, the two expected outputs are one and zero, respectively. The two expected outputs are zero and one if no plant were present. A plant was present for the data taken in the example above. The expected outputs are only ideal target outputs for the network. In service, the network's decision is indicated by the output that is numerically greater. If the first output is greater, then the system has detected a plant. If the second output is greater, then no plant has been detected.

The remainder of the line of data was added during development to help organize the data. This information is a code that indicates under what set of conditions this line of data was taken. For example, the code above indicates that this line of data was taken under sunny lighting, with the soil type Haskell, with dry soil, with 20 percent plant cover, and with a plant present in the viewing area. Appendix E gives more information on how to interpret the condition codes.

As mentioned above, the conditions varied for the collection of the training and testing data were soil type, lighting, soil moisture, and plant density. Each data set includes data from five different soil types, each of which has somewhat different coloring and texture. The ten soil samples used came from different locations around Oklahoma. The two different lighting conditions used for the tests were "sunny" and "cloudy." Soil moisture variations are "dry" and "wet." To take readings with wet soil, the soil was dampened using a spray bottle until the surface was saturated. Plant density variations are 0, 10, and 50 percent plant cover in the sensor viewing area. For

example, to obtain 50 percent plant cover, half the sensor viewing area was covered with uprooted bindweed.

Combinations of the conditions mentioned above would give 60 sets of data in both the training data set and in the testing data set--(5 soil types) x (2 lighting variations) x (2 soil moisture variations) x (3 plant densities) = 60 variations. Since the plant density variations are 0, 10, and 50 percent, these 60 sets have twice as many examples of "plant" conditions as they do "no plant" conditions. In order to eliminate this difference, 20 additional sets of readings were taken with no plant present, in both the training data set and the testing data set. These additional 20 sets of readings consist of the full combination of the remaining variables--five types of soil, two types of lighting, and two types of soil moisture. With the addition of this extra "no plant" data, the training and testing data sets each have 80 sets of data, 40 of which were taken with plants present, and 40 of which were taken with no plants present. This prevents any bias one way or the other for the neural network. If the neural network were trained with more examples with a plant present, then it might be more inclined to make that decision.

CHAPTER VII

SPRAYER UNIT SOFTWARE

Sprayer Unit Software Overview

The software in the sprayer unit computer provides the control algorithm for the sprayer unit. More specifically, the software controls the spray nozzle according to input conditions from the sensor. As the sprayer unit moves across a field, the objective of the software is to detect a plant under the sensor and activate the spray nozzle to apply herbicide to the plant. The code was developed in the C language and compiled to run on the Motorola MC68HC11A0 microprocessor in the weed sprayer computer. Following is an outline of the operation of the sprayer unit software.

1. Read red, green, and near infrared reflected radiation levels from the sensor.
2. Scale sensor readings for the neural network.
3. Pass scaled sensor readings through the neural network.
4. Determined, based on the scaled neural network outputs, whether or not a plant is present in the sensor viewing area.
5. Store "no plant" or "plant" decision at the rear of the nozzle state queue (FIFO).
6. Read earlier "no plant" or "plant" decision from the front of the nozzle state queue and update spray nozzle state (turn nozzle on or off) accordingly.
7. Loop to step 1.

The sprayer unit software uses a mathematical construct called a neural network to determine whether or not a plant is present in the sensor viewing area. The neural network was developed using a commercial software package, as described in Chapter VI, then implemented in C code for use in the sprayer unit software.

Originally, the sprayer unit software used a direct implementation of the neural network as created with the development package. This first implementation used exclusively floating point arithmetic and was found to execute too slowly for the weed sprayer

system. Consequently, two techniques were used to reduce the time required to make a forward pass through the neural network. The first technique was to implement the hyperbolic tangent activation function of each neural network node as a lookup table. The second technique was to convert all floating point arithmetic involved with the neural network to integer arithmetic. Sections of this chapter describe these techniques in greater detail.

Several of the sections of this chapter that explain the neural network implementation in the software are referring to the original network that was designed with the development package, which uses floating point arithmetic and calculates the hyperbolic tangent function for each node without using a lookup table. Such a discussion facilitates a more clear explanation of how the neural network operates. Each section of this chapter notes how it applies to each neural network implementation. Appendix C shows a program listing of the C code to implement the original floating point neural network. Appendix D shows a program listing of the C code for the final version of the sprayer unit software, with the neural network modifications implemented, and with code added to read the sensor and control the spray nozzle.

Reading Sensor Data

The software must obtain information from the sensor before it determines whether a plant is present in the sensor viewing area. The final version of the sprayer unit software (Appendix D) obtains data from the sensor using three of the eight analog-to-digital converters of the MC68HC11A0 microprocessor in the weed sprayer computer. The sensor supplies the three reflected radiation levels (red, green, and near infrared) as three voltages in the range of 0 to 5 volts. An analog-to-digital converter produces a number in the range of 0 to 255 (8 bits resolution) for each of the sensor voltages. The conversion takes place in hardware on the MC68HC11A0. The sprayer unit software need only begin the conversions, wait for the conversions to complete, then read the results from special result registers on the MC68HC11A0. The time required for these conversions is very small compared to the time required for other processing of the weed sprayer software. See the software code in Appendix C, Appendix D, as well as the *M68HC11 Reference Manual* for more information on the MC68HC11A0 analog-to-digital converters.

The code for the original floating point neural network implementation, as shown in Appendix C, does not read sensor data. The three variables that would normally hold the sensor readings are just assigned a set of readings from the training data set.

Neural Network Input Scaling

The sprayer unit software scales the sensor readings before passing them to the neural network. This section describes neural network input scaling in the original floating point implementation of the neural network. The Conversion from Floating Point to Integer section of this chapter describes the changes made to input scaling for the final software version.

Neural network input scaling is necessary because sensor readings fall within a different range of numbers than are useful to the neural network. Each sensor reading is an integer within the range 0 to 255. For the original floating point neural network, node outputs are always in the range -1 to 1 since they are calculated using the hyperbolic tangent (tanh) activation function of each node. Neural network input scaling maps the sensor readings to this range of -1 to 1 to be consistent with other parts of the neural network.

The scaling functions for each input (red, green, and near infrared) are slightly different and were formulated by the neural network development software. The mapping takes into account the minimum and maximum readings found in the neural network training data set for each sensor section. For example, the red sensor reading is scaled according to the following equation, which is very similar to a line of code in the original neural network software listing in Appendix C:

```
NODEOUT[0][0] = 2.0 * (REDIN - REDMIN) / (REDMAX - REDMIN) - 1.0,
```

where

NODEOUT[0][0] is an element of the two-dimensional array NODEOUT and contains the scaled sensor reading ready to be passed to the neural network.
REDIN = red sensor reading in the range 0-255.
REDMIN = minimum red sensor reading in the neural network training data set.
REDMAX = maximum red sensor reading in the neural network training data set.

This input scaling causes the minimum red sensor reading in the training data set to be mapped to -1 and the maximum red sensor reading in the training data set to be mapped to 1. Such a system helps to maximize resolution of the neural network inputs. The red and near infrared neural network inputs are scaled in a very similar way.

Basic Neural Network Implementation

The neural network is the part of the sprayer unit software that ultimately determines whether or not a plant is present in the sensor viewing area. The neural network structure is essentially the same for both the floating point and integer implementations. The Hyperbolic Tangent Lookup Table and the Conversion from Floating Point to Integer sections of this chapter describe the differences.

The neural network part of the software consists of a series of equations that involve the three scaled sensor readings (red, green, and near infrared) and the fixed weights developed as part of the network. The numerical outputs of the neural network are evaluated to determine whether the network has detected a plant in the sensor viewing area.

The trained neural network implemented in software in a sprayer unit is considered a fully connected, layered, feedforward network. Figure 7.1 shows a diagram of the sprayer unit neural network.

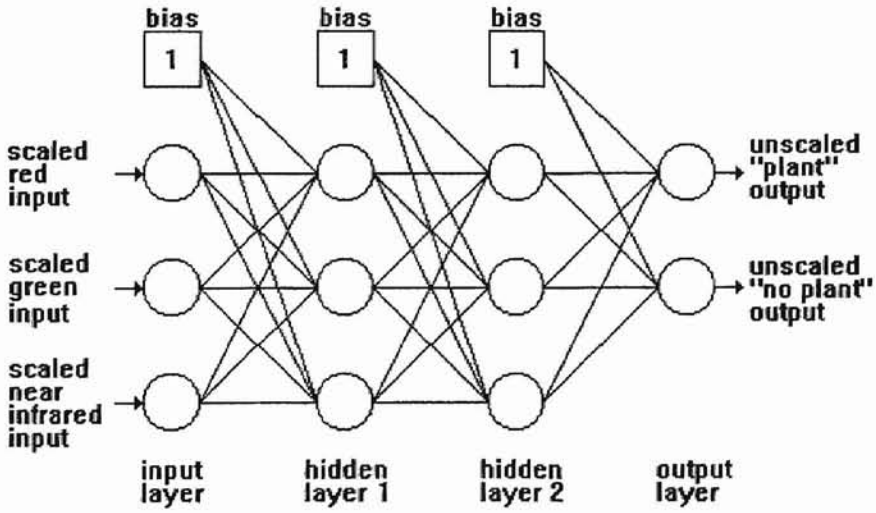


Figure 7.1: Sprayer unit neural network.

Information passes from left to right through the network. The network has an input layer consisting of three nodes; two hidden layers, each with three nodes; and an output layer consisting of two nodes. Each node in the hidden and output layers uses the hyperbolic tangent (tanh) as its activation function, or transfer function. No calculation is performed by the input layer nodes. The inputs to the network pass through the input layer unchanged. Each node in the hidden and output layers has a scaled bias input. The bias is unity and the bias input to each node has an associated weight, just like every other input to the node.

Appendix C shows a program listing of the C code to implement the original floating point neural network for the weed sprayer. This code performs a forward pass through the network. The network consists of two arrays. One array holds the output of each of the eight nodes. The second array holds the weights associated with each node. Each node can be considered a data structure that contains the weights on all the inputs to the node and the output of the node. The input nodes are not represented in the software, since they perform no operation on the data.

The sprayer unit software passes the scaled inputs through the network in a series of loops. The first hidden layer is processed first. The program calculates the weighted sum of all the inputs to each node. When working with the first hidden layer, these inputs are just the inputs to the network and the bias input. Next, the activation function (hyperbolic tangent) is applied to the weighted sum of each node and the result is stored as the output of the node. After each node in the first hidden layer is processed in this way, the program moves on to the next layer, which works similarly. The difference with the other layers is that the inputs to each node are now the outputs of the nodes in the previous layer, rather than the

inputs to the network. The outputs of the nodes in the final layer are the outputs of the neural network.

Figure 7.2 shows a block diagram of a neural network node for the original floating point neural network as developed.

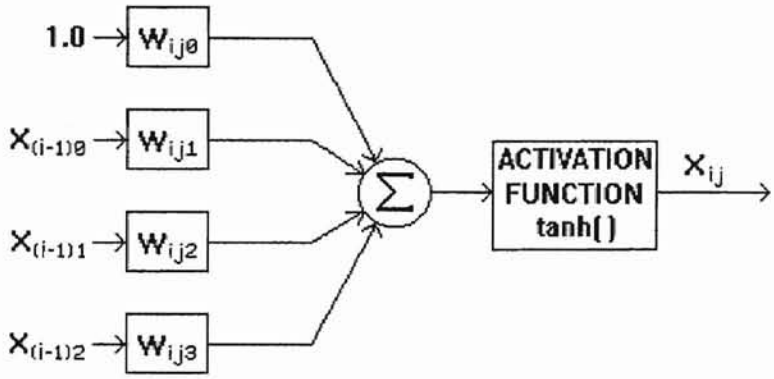


Figure 7.2: Block diagram of a sprayer unit neural network node.

On the left are the inputs from the previous layer. On the right is the node output. This model applies to every neural network node except for those in the input layer. The nodes in the input layer can be modeled by a multiplication by one.

For the original floating point neural network software implementation, the output of every neural network node, except those in the input layer, is given by the following equation:

$$X_{ij} = \tanh(1.0 * W_{ij0} + \Sigma_k [X_{(i-1)(k-1)} W_{ijk}]),$$

where

1.0 = the bias input to each node (unity) before being weighted.
i = layer number, starting with the first hidden layer
= 1, 2, ..., [number of layers - 1, including the input layer]
= the variable "layer" in the software.
j = node number in the current layer
= 0, 1, ..., [number of nodes in the current layer - 1]
= the variable "node" in the software.
k = node number in the previous layer
= weight number for the current node (bias weight is no. 0)
= 1, 2, ..., [number of nodes in the previous layer]
= the variable "i" in the software.

$X_{(i-1)(k-1)}$ = output of the (k-1)th node in the (i-1)th layer
= the array "nodeout[][]" in the software
= a real number.
W_{ijk} = the k-th weight for the j-th node in the i-th layer
= the array "weight[][][]" in the software
= a real number.

The equation for the output of each node differs from this in the integer implementation of the neural network in the final version of the sprayer unit software. The differences are discussed in the Hyperbolic Tangent Lookup Table and the Conversion from Floating Point to Integer sections of this chapter.

Neural Network Output Scaling

The neural network output scaling equations were produced by the neural network development software in order to make the actual network outputs compatible with the ideal expected outputs specified by the user in the training and testing data sets. Neural network output scaling was required for this project only during the neural network development process, when it was useful to compare scaled neural network outputs with the ideal expected outputs. The original floating point neural network implementation incorporates neural network output scaling. The scaling is not necessary in the final implementation of the neural network in the sprayer software since the relative magnitudes of the two outputs is what determines the network's decision. Neural network output scaling here would add unnecessary software execution time.

The two unscaled outputs of the neural network are the results of calculations in the two nodes in the output layer. Since, in the floating point neural network implementation, these outputs are calculated using the hyperbolic tangent function, they fall within the range -1 to 1. The ideal expected outputs of the neural network are (0, 1) when no plant is present and (1, 0) when a plant is present. Thus each scaled output is expected to fall within the range 0 to 1. Neural network output scaling maps each actual neural network output from the range -1 to 1 to the range 0 to 1. This allows the designer to see how close the neural network outputs are to ideal.

The scaling is identical for the two neural network outputs. The "plant" output is scaled according to the following equation, which is taken directly from the original floating point neural network software listing in Appendix C:

```
OUTPUT1 = (NODEOUT[3][0] + 0.8) / 1.6,
```

where

OUTPUT1 is the scaled "plant" neural network output.
NODEOUT[3][0] is an element of the two-dimensional array NODEOUT and contains the unscaled "plant" neural network output.

Note that this scaling actually maps a neural network output of -0.8 to 0 and maps an output of 0.8 to 1, instead of mapping -1 (the lower bound of the tanh function) to 0 and 1 (the upper bound of the tanh function) to 1. The development software chooses this mapping so that hitting the ideal outputs during neural network training is actually possible. The latter mapping is not used because the hyperbolic tangent result cannot actually reach -1 or 1, and so the ideal expected output could never be reached. Tanh(x) approaches -1 for large negative values of x and approaches 1 for large positive values of x.

Interpreting Neural Network Outputs

After the sprayer unit software passes sensor readings through the neural network, the network outputs are evaluated to determine whether or not a plant has been detected in the viewing area of the sensor. The neural network has two outputs, one corresponding to a "plant" decision and one corresponding to a "no plant" decision. The output that is numerically greater indicates the neural network's decision. This holds true even in the final sprayer unit software, where output scaling has been removed, and integer arithmetic is used for the neural network. With the original floating point implementation, the ideal scaled neural network outputs are (1, 0) with a plant present and (0, 1) with no plant present. The Sprayer Unit Software Performance Results section of this chapter shows some typical scaled and unscaled outputs for the two neural network implementations.

Hyperbolic Tangent Lookup Table

The final version of the sprayer unit software uses a lookup table to implement the hyperbolic tangent (tanh) transfer function of each neural network node. The lookup table allows software to "look up" the result of a hyperbolic tangent calculation in a table in memory rather than calculate the result. Such a system allows the sprayer unit software to use much less processing time than the original floating point version and allows the software to run at a higher frequency.

The original C code to implement the neural network included a "tanh(x)" C function for each node. The C compiler for the MC68HC11A0 microprocessor converted each of these C functions to a series approximation to run on the MC68HC11A0. Since the MC68HC11A0 in the weed sprayer computer can only manipulate 8-bit and 16-bit integer numbers, these series approximations required high overhead in processing time. With the lookup table, the evaluation of the hyperbolic tangent function amounts only to a few memory accesses, which requires much less time to execute. The disadvantage with this method is that some error results in the function evaluation, since the hyperbolic tangent outcomes are limited to the 256 values stored in the table. The Sprayer Unit Software Performance Results section of this chapter quantifies the advantages and disadvantages of using the lookup table.

The hyperbolic tangent lookup table was originally developed as a true tanh function, before being scaled up for the integer version of the software. The tanh(x) function is equal to the following expression:

$$\tanh(x) = (e^{x} - e^{-x}) / (e^{x} + e^{-x})$$

Figure 7.3 shows a plot of the hyperbolic tangent function. The original table could access tanh(x) for x in the interval -2 to 2, inclusive. This range was chosen as a compromise between good resolution in the table and wasted space in the table. Most of the "active" part of the function falls within the -2 to 2 region. Accepting a wider range of inputs to the table would mean lower resolution in this "active" region and increased error. Tests run by passing typical sensor readings through the neural network have shown that the bounds of the lookup table, covering this range of inputs, are rarely exceeded.

The lookup table used in the final version of the sprayer unit software does not implement a true hyperbolic tangent function, since all arithmetic has been converted to use only integers. The lookup table is a scaled version of the hyperbolic tangent function. See the Conversion from Floating Point to Integer section of this chapter for more information.

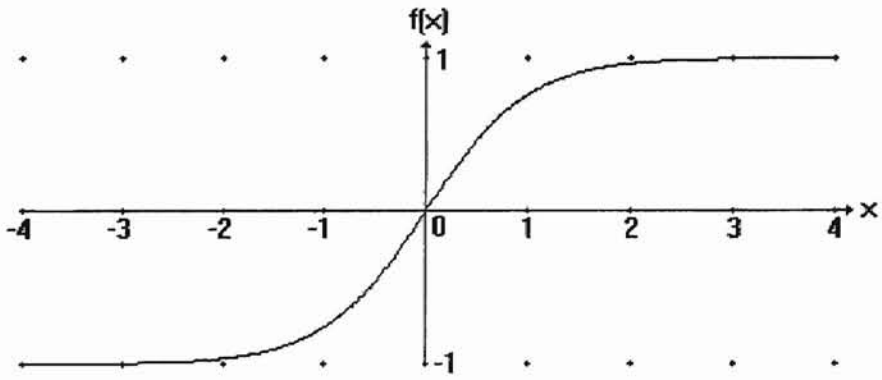


Figure 7.3: The hyperbolic tangent (tanh) function.

Conversion from Floating Point to Integer

The final sprayer unit software uses exclusively integer arithmetic (at the C code level) to perform all calculations. This requires much less processing time than the original version, which had floating point C instructions. Since many calculations are made in the neural network, conversion to integers allows the sprayer unit software to run at a much higher frequency. Appendix C shows a program listing of the C code for the original floating point version of the neural network. Appendix D shows a program listing for the final integer version of the code.

The MC68HC11A0 microprocessor used in the weed sprayer computer supports 8-bit and limited 16-bit integer arithmetic. This means that floating point C instructions require a series of many integer operations on the MC68HC11A0. Integer C instructions using the "long int" type, which uses 32-bit signed integers, require significantly fewer MC68HC11A0 integer operations.

Conversion of the floating point algorithm to integers essentially consisted of scaling up all numbers to eliminate the fractional parts. This was a compromise between losing precision and using integers in a range that can be manipulated quickly by the MC68HC11A0 microprocessor. The larger the number of significant digits used, the more precision the calculations carry. On the other hand, larger numbers require more bits for storage, thus more processing time to manipulate. Following is a list of specific changes made to convert the original floating point software to integers:

1. For each neural network node, each weight, except for the weight on the bias input, was multiplied by 10^4 and rounded to the nearest integer.
   EXAMPLE: The weight 1.906368 was converted to 19064. This is the second weight of the first node in the first hidden layer.

2. For each neural network node, the weight on the bias input was multiplied by 10^8. The weight on the bias input to each node is treated differently because the unweighted bias input is always unity, and is not the output of another node, which would already be scaled up by 10^4.
   During code development for the integer version of the software, bias weights were inadvertently rounded to the nearest 10000. This results in unnecessary loss of precision. In most cases, two more significant figures could have been kept.
   EXAMPLE: The weight -0.127484 was converted to -12750000. This is the first weight of the first node in the first hidden layer.

3. Neural network input scaling equations were multiplied by 10^4.
   EXAMPLE:
   The following equation,
   ```
   nodeout[0][0] = (redin-redmin)/(redmax-redmin)*2.0-1.0;
   ```
   was converted to
   ```
   nodeout[0][0] = 20000*(redin-redmin)/(redmax-redmin)-10000;
   ```

4. The modified hyperbolic tangent (tanh) lookup table contains 256 entries that are actually tanh results scaled up by 10^4. In the integer software version, neural network node weighted sums, which are integers in the range of $-2x10^8$ to $2x10^8$, are converted to integers in the range 0 to 255 to access the lookup table. This table represents the activation function, or transfer function, of each node. The lookup table evaluates the following function:
   $y = 10^4 * \tanh(x / 10^8)$.
   See the Hyperbolic Tangent Lookup Table section of this chapter for more information.

Disadvantages of the conversion from floating point to integers are that the neural network implementation is not so straightforward and is more difficult to understand. Another disadvantage is that some precision is lost in all calculations due to the rounding that occurs in the conversion. This results in error at the neural network outputs, when compared to the neural network as originally developed. The Sprayer Unit Software Performance Results section quantifies the

advantages and disadvantages of converting all floating point arithmetic to integer arithmetic in the sprayer unit software.

Spray Nozzle Timing

Software timing must ensure that weeds detected on the ground are properly targeted by the moving sprayer unit. As a sprayer unit moves over a field, the sensor physically leads the spray nozzle on the boom. This is necessary so that, when a plant is detected, there is time for the nozzle to open and for the liquid to travel to the ground before the plant passes. The distance that the sensor leads the nozzle is far enough that some time must elapse between the time that a reading is taken and the time that the nozzle is activated. This allows the delay time to be adjusted in software to match a particular tractor speed. The software cannot merely take a reading and then pause until the proper time to turn on the nozzle because ground would be passed over in the mean time, and a weed might be missed. Instead, the software stores nozzle state information in a queue (FIFO) for sending to the spray nozzle at a later time. Using such a queue does not significantly affect software execution speed, but does cause the designed delay between each sensor reading and the corresponding spray nozzle updates.

During each iteration in the sprayer unit software, after sensor readings have been passed through the neural network and a decision is made whether a plant is present in the viewing area or not, the decision is placed at the rear of the nozzle state queue. The item at the front of the queue is then retrieved and the spray nozzle is set accordingly. The spray nozzle remains in this state, either on or off, until the next software iteration.

The nozzle state queue keeps a history of the past 19 neural network decisions ("no plant" or "plant"), thus it has a length of 19 items. The length of the queue was matched to a particular ground speed at which the tractor must run in order for the system to target the weeds properly. In order to match the queue length to a particular

ground speed, a rough calculation was first made. This involved calculating the required queue length based on the known speed of the tractor in a certain gear at a certain engine RPM. The equation used that relates these quantities is the following:

$$L = (f * d / v) - t,$$

where

L = length of the nozzle state queue (no. of items).
f = program execution frequency (determined experimentally).
d = distance (in direction of travel) from center of sensor to center of spray nozzle.
  = 38.0 in.
v = tractor ground speed.
t = elapsed time from computer sending signal to turn on spray nozzle until fluid strikes the ground (determined experimentally). Measured at 25 psi and 24 in. travel height.
  = 0.107 seconds.

Field tests showed that the initial calculation for the required queue length was incorrect and the spray did not hit the targeted plants. This error was probably due primarily to an incorrect figure being used for the travel time of the fluid from the spray nozzle to the ground. To correct this problem the tractor speed was matched to the fixed software timing using trial and error. The tractor provides a digital RPM display, so the required speed could be repeated each time the system was used.

Sprayer Unit Software Performance Results

Software execution speed and software computational accuracy were the two main software performance concerns for this project. Both of the main techniques used to increase execution speed of the sprayer unit software cause the disadvantage of decreased computational accuracy as compared to the original neural network as it was developed. Both techniques result in some round-off error at the neural network outputs, which may result in incorrect software decisions ("plant" or "no plant") for "marginal" cases. Speed improvements are significant and errors in neural network outputs are small and were deemed acceptable.

Software execution speed is important because it affects the maximum ground speed that the sprayer system can be driven across a field. A good measure of software execution speed is the length of time required for the software to make one cycle through the code. This is effectively the time required for one "plant"/"no plant" decision. Table 7.1 compares the software execution time for a full cycle through the code, during different stages of software development. Execution time was reduced by approximately a factor of four using the two techniques described in this chapter. Based on the best execution time and on the length, or dimension in the direction of travel, of the sensor viewing area, the maximum ground speed for the sprayer system is about 2.15 miles per hour.

Table 7.1: Software execution times at different stages of development.

TYPE OF ARITHMETIC	ACTIVATION FUNCTION	EXECUTION TIME (sec)
floating point	tanh(x)	0.29
floating point	tanh(x) lookup table	0.13
integer	tanh(x) lookup table	0.070

The 256-location hyperbolic tangent lookup table used to implement the activation function of each neural network node rounds calculations of the function to a value listed in the table. Conversion, from floating point to integer, of all calculations made in the neural network forward pass results in round-off error since the integers used do not generally have as many significant figures as the original floating point numbers did. Examples of this can be seen by comparing the array of neural network weights in the two software versions (Appendix C and Appendix D). The weights in the original floating point version generally have 6 or 7 significant figures, while the integer weights in the final integer version have 4 or 5 significant figures (for example, 1.906368 vs. 19064). The magnitude of the integers used is limited by the C language variable types that the compiler supports. Variable space (resolution) was not optimized in the sprayer unit software when the conversion was made to integers. The conversion was merely done in a simple way, as described in the Conversion from Floating Point to Integers section of this chapter. The resulting errors at the neural network outputs were deemed acceptable.

Table 7.2 shows calculated neural network outputs for the two extreme cases of the software, in response to the same neural network inputs. The six sets of inputs are actual sensor readings from the testing data set. A comparison of the "SCALED NEURAL NETWORK OUTPUTS" columns of the two sections shows examples of the errors that result at the neural network outputs due to the speed improvement modifications. Notice that the differences between these two sections of the tables are small. The scaled neural network outputs for the first section of the table were generated using the methods described in the Neural Network Output Scaling section of this chapter. The scaled outputs for the second section were generated using very similar linear equations.

Table 7.2: Comparison of neural network outputs before and after speed improvement techniques were applied.

FLOATING POINT ARITHMETIC WITH ACTUAL TANH(X) FUNCTION.

NEURAL NETWORK INPUTS			UNSCALED NEURAL NETWORK OUTPUTS		SCALED NEURAL NETWORK OUTPUTS	
RED	GRN	NIR	OUTPUT 1	OUTPUT 2	OUTPUT 1	OUTPUT 2
113	86	84	-0.803878	0.803503	-0.002424	1.002190
80	67	80	0.796874	-0.796682	0.998046	0.002074
54	34	43	-0.405208	0.406352	0.246745	0.753970
46	37	39	0.274699	-0.275862	0.671687	0.327586
36	29	31	0.669625	-0.670443	0.918516	0.080973
84	78	113	0.824791	-0.823956	1.015494	-0.014972

INTEGER ARITHMETIC WITH INTEGER LOOKUP TABLE FOR TANH(X).

NEURAL NETWORK INPUTS			UNSCALED NEURAL NETWORK OUTPUTS		SCALED NEURAL NETWORK OUTPUTS	
RED	GRN	NIR	OUTPUT 1	OUTPUT 2	OUTPUT 1	OUTPUT 2
113	86	84	-8026	7969	-0.001625	0.998062
80	67	80	7911	-7969	0.994438	0.001937
54	34	43	-4194	4064	0.237875	0.754000
46	37	39	2678	-2823	0.667375	0.323563
36	29	31	6683	-6769	0.917687	0.076937
84	78	113	8187	-8187	1.011688	-0.011688

Note that in the final version of the sprayer unit software, with the integer arithmetic and lookup table, neural network outputs are not actually scaled. They are only scaled here in the table for a comparison with the previous network implementation.

CHAPTER VIII

SYSTEM PERFORMANCE RESULTS

Sensor Prototype Comparison

During sensor development, tests were conducted to determine whether or not sensor performance would be consistent when several sensors were constructed using the same parts. In other words, the following questions was investigated: "When additional sensors are constructed, will they behave sufficiently similar to the original prototype?" Early investigation into this matter was based on two sensors built using the same construction techniques and parts. The two prototypes, called prototype 2 and prototype 3, were of the same construction as the sensor described in Chapter IV.

The output voltages of the two prototypes were compared under nearly identical conditions. Using an outdoor test stand, each sens was, in turn, suspended above a soil surface and the red, green, and near infrared output voltages were recorded. These measurements sho that there was very close correspondence between the respective outp of the two sensors. The red output voltage of prototype 2 was very close to the red output voltage of prototype 3. Results were simila for the green and near infrared voltages. Comparison of these two original sensors showed that their performance was very similar and repeatable performance for additional sensors was indicated.

Field Sprayer Unit Comparison

A qualitative comparison of the plant detection capabilities of the five sprayer units in the experimental sprayer system showed significant performance differences. The five sensors used on these five sprayer units included the two original sensor prototypes (prototypes 2 and 3), in addition to three other sensors that were later constructed using the same parts and techniques. The qualitativ tests were run with the five complete sprayer units mounted on the spray boom, with each suspended above natural soil in a field. Each sprayer unit ran software that implemented the original floating point version of the neural network. The tractor and spray boom were stationary for these tests. The performance of each sprayer unit was evaluated by holding parts of a bindweed plant near the ground directl under the sensor, in the viewing area of the unit, and observing the ability of the unit to detect the plant.

During the field tests, each sprayer unit worked correctly in that plant material in the viewing area caused activation, or a turnin on of the spray nozzle. However, different amounts of plant cover wer required with each sensor to cause activation. One of the sensors was much more "sensitive" than the others and could be activated consistently with an estimated five percent plant cover in the viewing area. This unit also occasionally activated when no plant was present as lighting conditions changed due to cloud cover. The other sprayer units required a higher plant cover. In some cases, an estimated 30 percent cover was required to cause plant detection. Investigations into the cause(s) of the sprayer unit variations are described in othe sections of this chapter.

Sensor Output Comparison

The output voltages of the sensors on the five sprayer units of the experimental system were compared under nearly identical conditions. These tests were carried out in an attempt to find the cause of the pronounced variation in the ability of the five sprayer units to detect plants. For these tests, the sprayer units were suspended above the ground by the spray boom. Soil was spread out on a platform on the ground, which could be placed, in turn, under each of the five sensor of the sprayer units. For each sensor, the three output voltages (red, green, and near infrared) were measured, with the soil covering the entire viewing area of the sensor during the measurement. These measurements showed significant variations in the respective output voltages of the sensors, as shown in Table 8.1.

Table 8.1: Output voltages of the five sensors under nearly identical conditions. Conditions: Fort Cobb soil, full sun, dry soil, zero plant cover.

SENSOR	SENSOR OUTPUT VOLTAGES (volts)		
	RED	GREEN	NEAR INFRARED
1 (prototype 2)	1.28	0.685	0.905
2	1.30	0.652	0.900
3	1.34	0.777	0.934
4	1.59	0.926	1.270
5 (prototype 3)	1.37	0.835	0.986

For consistent performance from sensor to sensor, the output voltages in each band would be very close to one another. For example, the red outputs for all five sensors would be nearly the same. This is clearly not the case. Variations are significant in all three sensor bands. Note that the output voltages of sensor 4 are much higher than those of the other sensors. Output voltages measured running these same tests, but using one of the other soil types, showed similar variations.

Variations in sensor output voltages were measured using an indoor test similar to the one described above. In these tests each

sensor was mounted, in turn, on a test stand and suspended above a fixed white surface. An ordinary incandescent lamp held in a fixed position was used as the light source for these tests. Results were comparable to those above in that there were wide variations between sensors, and sensor 4 showed the highest readings.

The variations measured in sensor output voltages during the outdoor and indoor tests are believed to be sufficient to cause the variations in sprayer unit performance that were observed in the fiel performance tests. These field tests are described in the section entitled Field Sprayer Unit Comparison in this chapter. Computer hardware and software for all five units was essentially identical an is unlikely to cause performance differences between units.

The following are believed to be the most likely causes of the variations in sensors: differences in the photodiode responses, optical differences in the filters used on the photodiodes, and differences in the optical properties of the interior of the sensor housings. Each photodiode likely has a different response when expos to the same lighting. Since each filter was custom cut from a larger piece of material, the filters are not all identical in shape. The textures of the cut surfaces of the filters also differ. The fact tha the three readings for sensor 4 were all significantly higher than the readings of the other sensors indicates that possibly light is reflected better inside this sensor housing than the others. Other sensor components, including the resistors, operational amplifiers, ar the sensor circuit board masks, were eliminated as likely sources of the major output voltage variations observed.

The Effects of Time of Day

Sprayer unit performance is affected by the time of day of operation. Tests show that the ability of the system to detect plants is degraded in the early morning and late evening hours. In general, a higher plant cover is required to activate a sprayer unit during these times. This problem is likely due to the fact that the intensities of the different colors in the sunlight reaching the surface of the earth are different in the morning and evening hours (Sagan 1994). At these times of the day, light from the sun must travel a greater distance through the atmosphere before reaching the ground. The colors with shorter wavelengths, such as blue and violet, are scattered more efficiently by the atmosphere, thus the intensities of these colors are reduced when the sun is low on the horizon. Since the sprayer unit relies on the intensities of the different colors of light in its plant detection algorithm, performance is affected when the relative intensities change.

The Effects of Soil Moisture

Although wet soil was taken into account during neural network training, tests show that sprayer unit performance is degraded with some soil types when wet. Even using sensor prototype 3, the sensor that was used to collect the neural network training and testing data, plant detection performance was poor under certain conditions with wet soil. In some cases, false plant detections occurred when no plant was present in the viewing area. The causes of such performance are unknown.

The Effects of Soil Texture

Tests show that the texture of soil that the sprayer unit encounters affects its performance. Although exposure of the neural network to several different soil types during training prepares the sprayer unit for different soil colorings, different soil textures are not taken into account. Soil texture encountered in the field during sprayer unit operation was distinctly different than that used during the neural network training portion of system development. The major differences observed in the natural soil in the field are in the following areas: soil smoothness, size of clods, and presence of dead plant material.

Due to the effects of water and wind erosion, the soil on the ground in the field was much more smooth than that placed on the test stand for the acquisition of neural network training and testing data. The smoother, weathered soil in the field appears to reflect light better than the more broken-up soil used on the test stand.

The soil in the field had much larger clods, which tend to cause shadows. These shadows may adversely affect sprayer unit performance, as described in the section of this chapter entitled The Effects of Shadows.

Dead plant material, or crop residue, in the field on the ground surface reflects light differently than soil, and may adversely affect sprayer unit performance when in the viewing area of the sensor. However, tests indicate (Nitsch 1991) that some weathered crop residues have reflectivity curves that are very similar to that of soil. This is due to the fact that chlorophyll is largely absent in this long-dead plant material. Based on this information, crop residue may slightly affect sprayer unit performance, but is unlikely to cause major problems.

The Effects of Shadows

Shadows in the sensor viewing area adversely affect sprayer unit performance. Tests show that, with large shadows in the sensor viewing area, a greater percentage of plant cover is required to cause the sprayer unit to detect the plant. The effects of shadows decrease with the shadow size. The inability of the neural network used in the plant detection algorithm to properly classify shadowed fields of view can be explained by the fact that shadows were not taken into account during neural network training. The effect shadows have on the sensor readings is unknown, but since shadowed areas reflect light differently than areas directly illuminated by the sun, this situation might be expected to affect performance. An improved plant detection algorithm might result from adding shadowed plant and soil settings to the neural network training process.

CHAPTER IX

CONCLUSIONS AND RECOMMENDATIONS

Overall System Performance

Although the basic plant detection and spraying methods presented in this thesis showed potential, the implementation of the experimental system described here showed significant limitations. The problems were primarily in the area of plant detection, and are likely due to the neural network based plant detection algorithm and the sensors. Although the weed sprayer system worked moderately well and was able to target weeds under some conditions, plant detection performance suffered because of variations in the following areas: cloud cover, time of day, soil moisture, soil texture, and shadows. The neural network training and testing was likely insufficient in providing a plant detection algorithm capable of adapting to the wide variation of conditions encountered outdoors in an agricultural setting. Additional problems with wide performance variation among the sensors was observed. The weed targeting methods used in the sprayer unit software worked well. However, software execution time of the math-intensive plant detection algorithm on the available computer hardware limited tractor speeds. Otherwise, the computer hardware worked well for the job.

Sensor

The three-band optical sensor used in the sprayer unit provided measurements of reflected radiation that proved useful in the detection of bindweed (Convolvulus arvensis) under a wide range of environmental conditions. This is evident in the success of the neural network development process, where only one sensor was utilized. However, construction and testing of additional sensors showed significant performance variation. The performance differences in the sensors were great enough to render some sprayer units ineffective for the task of plant detection. A method of sensor calibration to compensate for electrical component variations is recommended. In addition, the sensor parts affecting the optical properties of the device should be very closely matched among sensors.

Neural Network

Neural network development for the sprayer units was successful in showing that a neural network could be created to recognize bindweed (Convolvulus arvensis) under a wide range of conditions using one of the three-band optical sensors developed for this project. Using a single sensor prototype to collect training and testing data, the neural network that was developed correctly classified all settings in the testing data. Implementation of the neural network in a plant detection algorithm using additional sensors under new field conditions revealed significant problems. The neural network was unable to effectively adapt to variation in the sensors or to new variations in cloud cover, time of day, soil moisture, soil texture, and shadows. If a neural network is to be developed that can adapt to such a wide range of conditions, it is recommended that much more attention be given to the neural network development process. The collection of training and testing data under a much wider range of conditions would likely be required. This would significantly increase the time required for collection of field data and would likely increase the time required for each neural network training and testing cycle during the development process.

Sprayer Unit Software

The sprayer unit software implements the plant detection algorithm of the unit. Although the neural network based plant detection algorithm used in the sprayer units showed good results und some operating conditions, it did not prove successful in adapting to wide variety of field situations, as discussed in the Neural Network section of this chapter. In addition, the ground speed of the tracto carrying the spray boom and sprayer units was severely limited by the execution speed of the software running the plant detection algorithm In a practical system, a much simpler algorithm would likely be required in order to utilize the same relatively inexpensive computer hardware. A simpler algorithm may exist that is just as effective, perhaps using the sensor readings to calculate one of the vegetation indices discussed in Chapter II. Using such a simple calculation instead of the neural network would also eliminate the need for the time-consuming process of collection training and testing data for th neural network development process. The timing system used for weed targeting in the sprayer unit, as discussed in the Spray Nozzle Timin section of Chapter VII, worked effectively.

Computer

The Dearborn Network Analysis Tool (DNAT), based on the Motorola MC68HC11A0 microprocessor, was relatively inexpensive and easy to use, but limited execution speed of the plant detection algorithm. This algorithm, which involves a neural network, requires a large number of mathematical calculations. Programming techniques were used to reduce execution times and achieve a software execution rate of about 14 decisions per second. Still further speed increases would likely be required in a practical weed sprayer system. This speed-up might involve using either a much simpler plant detection algorithm, or using faster computer hardware. Another option is to incorporate larger amounts of information in lookup table form, although this would likely require additional computer memory. The DNAT computer was easily programmed and provided sufficient I/O (input/output) capabilities for the sprayer unit.

BIBLIOGRAPHY

Chrysler Network Analysis Tool--CNAT User's Manual, version 1.2. Dearborn Group, Inc. 1992.

Cicone, Richard C. and Michael D. Metzler. 1984. Comparison of Landsat MSS, Nimbus-7 CZCS, and NOAA-7 AVHRR features for land-use analysis. *Remote sensing of environment*. New York: Elsevier Science Publishing Co., Inc.

Crist, E. P. and R. J. Kauth. 1986. The tasseled cap de-mystified. *Photogrammetric engineering and remote sensing*, Jan. 1986. American Society for Photogrammetry and Remote Sensing.

Shropshire, Geoffrey J., Kenneth Von Bargen, and David A. Mortensen. 1990. Optical reflectance sensor for detecting plants. *Proceedings of the SPIE--The International Society for Optical Engineering*, vol. 1379.

Felton, W. L. and K. R. McCloy. 1992. Spot spraying. *Agricultural engineering*, Nov. 1992.

Gupta, R. K. 1993. Comparative study of AVHRR ratio vegetation inde and normalized difference vegetation index in district level agricultural monitoring. *Remote sensing*, vol. 14, no. 1.

Kauth, R. J. and G. S. Thomas. 1976. The tasselled cap--A graphic description of the spectral-temporal development of agricultura crops as seen by Landsat. Environmental Research Institute of Michigan.

M68HC11 Reference Manual. 1991. Motorola, Inc.

Neural Computing. 1991. Pittsburg, PA: NeuralWare, Inc.

Nitsch, B. B., K. Von Bargen, G. E. Meyer, D. A. Mortensen. 1991. Visible and near infrared plant, soil and crop residue reflectivity for weed sensor design. Meeting presentation for the American Society of Agricultural Engineers, paper no. 91300

Perry, Charles R. Jr. and Lyle F. Lautenschlager. 1984. Functional equivalence of spectral vegetation indices. *Remote sensing of environment*. New York: Elsevier Science Publishing Co., Inc.

Rich, Elaine and Kevin Knight. 1991. *Artificial intelligence*, 2nd e New York: McGraw-Hill, Inc.

Sagan, Carl. 1994. *Pale Blue Dot*. New York: Random House, Inc.

Shropshire, Geoffrey J., Kenneth Von Bargen, and David A. Mortensen. 1990. Optical reflectance sensor for detecting plants. *Proceedings of the SPIE--The International Society for Optical Engineering*, vol. 1379.

Stone, Marvin L., Embedded Neural Networks in Real Time Controls. *SA Technical Paper Series*, paper 941067.

Wiegand, Craig L. and Arthur J. Richardson. 1992. Relating spectral observations to yield. United States Department of Agriculture

APPENDIX A

SENSOR OPTICAL GEOMETRY DIAGRAMS AND CALCULATIONS

Figures A-1, A-2, A-3 show the sensor optical geometry and the geometry of the sensor viewing area.

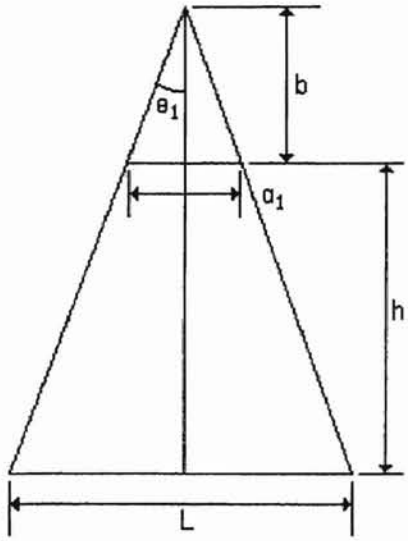


Figure A-1: Sensor optical geometry–side view.

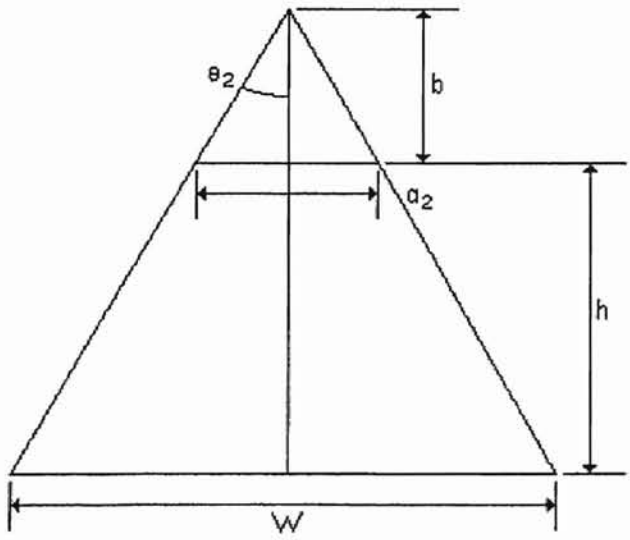


Figure A-2: Sensor optical geometry–front view.

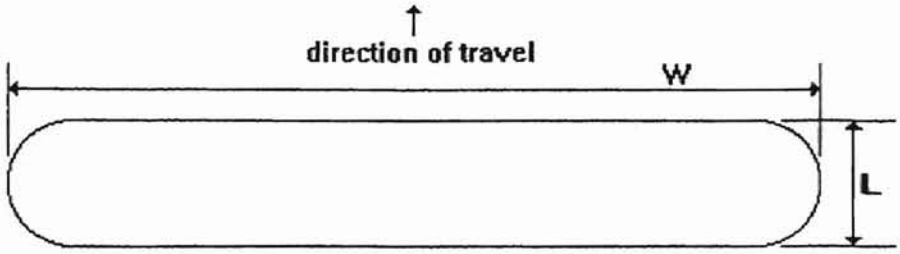


Figure A-3: Sensor viewing area.

Definitions for sensor optical geometry calculations and diagrams.

a_1 = Length (in direction of travel) of aperature in sensor aperature plate.
= 0.25 in.

a_2 = Width (perpendicular to direction of travel) of aperature in sensor aperature plate.
= 2.125 in.

b = Distance from lower photodiode surface to lower surface of sensor aperature plate.
= 2.5 in.

h = Distance from lower surface of sensor aperature plate to the ground surface.
= 24 in.

L = Length (in direction of travel) of sensor viewing area on the ground surface.
= 2.65 in.

W = Width (perpendicular to direction of travel) of sensor viewing area on the ground surface.
= 22.5 in.

Sensor optical geometry calculations:

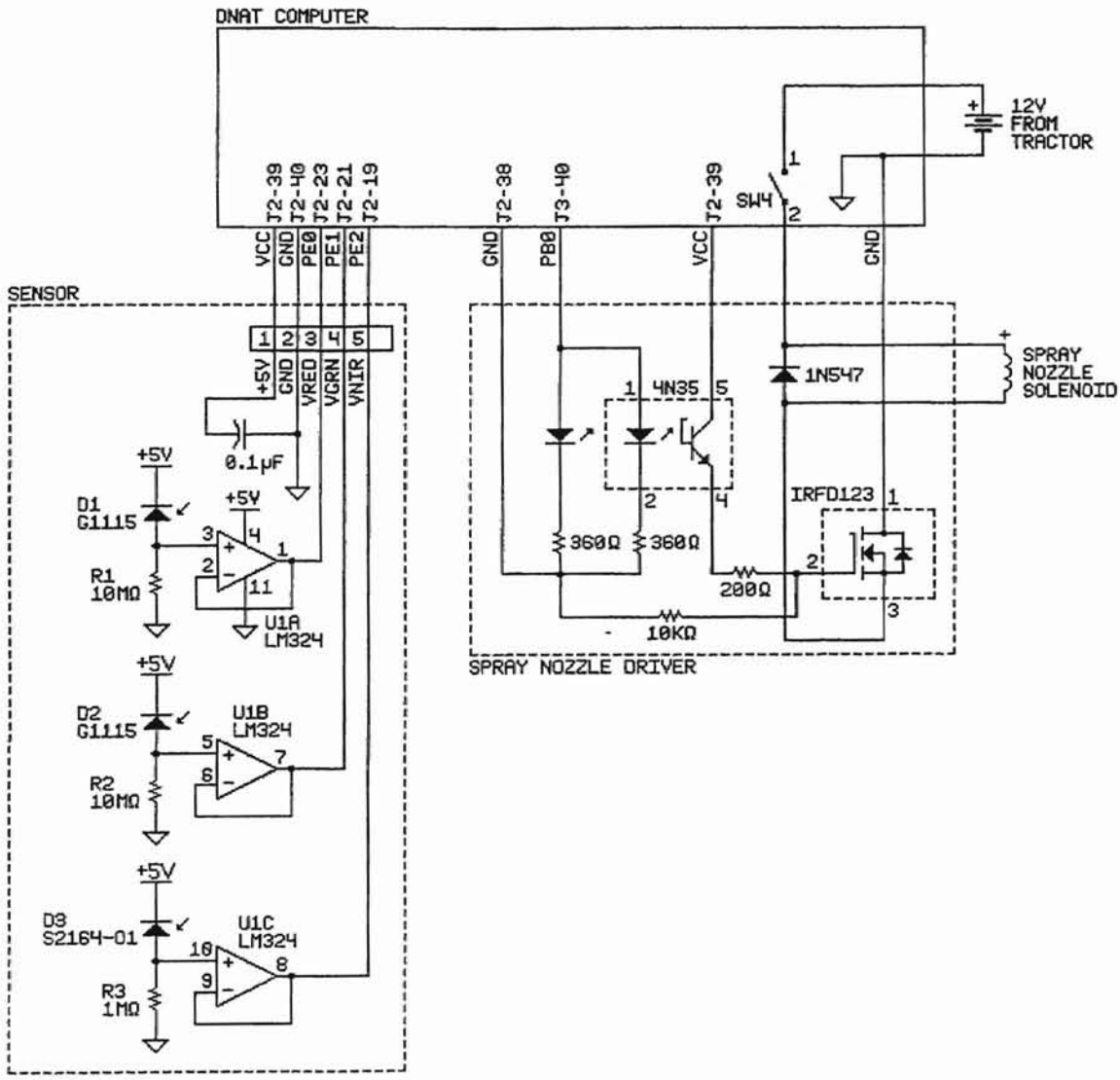
$(L/2)/(b+h) = (a_1/2)/b$
$L/(b+h) = a_1/b$
$L = a_1(b+h)/b$
$L = (0.25)(2.5+24)/2.5$
$L = 2.65$ in.

$(W/2)/(b+h) = (a_2/2)/b$
$W/(b+h) = a_{2/b}$
$W = a_2(b+h)/b$
$W = (2.125)(2.5+24)/2.5$
$W = 22.5$ in.

APPENDIX B

SPRAYER UNIT SCHEMATIC DIAGRAM

SPRAYER UNIT SCHEMATIC DIAGRAM



APPENDIX C

PROGRAM LISTING OF ORIGINAL FLOATING POINT NEURAL NETWORK SOFTWARE

```
/*  Ronnie Evans  */
/*  This program is a modified form of SPRAY10.C.                    */
/*  Intended for compilation by Introl C compiler and execution      */
/*  on the 68HC11 computer.                                          */
/*  Incorporates new weights obtained from training on data from     */
/*  prototype 3.
/*  This program takes red, blue-green, and nir sensor readings,     */
/*  runs them through a neural network and decides whether there     */
/*  is a plant present or not.                                       */
/*  The three inputs are assigned in 'main'.                         */
/*  The algorithm for finding the two network outputs (called        */
/*  out1 and out2 below) is the same as that used by NeuralWare      */
/*  on a forward pass (test) of the network.                         */

#include <math.h>
#include <stdio.h>
#include <string.h>
#include "h11reg.h"
#include "terminal.h"

/* Following are min. and max. readings in the training data set. */
/* They are used later to scale the inputs to the neural network. */
#define redmin 28.0      /* Minimum red reading in training data set. */
#define redmax 172.0     /* Maximum red. */
#define greenmin 21.0    /* Minimum green (blue-green). */
#define greenmax 114.0   /* Maximum green. */
#define nirmin 15.0      /* Minimum near infrared (NIR). */
#define nirmax 170.0     /* Maximum near infrared. */

#define DONOTHING
#define NUMLAYERS 4  /* Number of layers in neural network, */
   /* including input layer. */
int layer, node, i;
float sum, output1, output2;
int numnodes[NUMLAYERS] = {3, 3, 3, 2};
   /* numnodes array contains number of nodes in each layer,  */
   /* including input layer.  */
float nodeout[NUMLAYERS][4];
   /* nodeout array contains node outputs. */
   /* indices are [layer#][node#]. */
float weight[NUMLAYERS][3][4] =
{{{  0,          0,           0,          0         }   /* nothing */
 },
 {{ -0.127484,  1.906368,   0.146913, -2.456775 }, /* hidden layer 1 */
  {  0.225026, -10.56199,  -1.797332, 13.357224 },
  { -0.296284,  4.162867,   0.346963, -5.187379 }
 },
 {{ -0.112190,  0.271015,  -1.034428,  0.586344 }, /* hidden layer 2 */
  {  0.155440, -0.430254,   1.098036, -0.768516 },
  { -0.050510, -0.087746,   1.971593, -0.302194 }
 },
 {{  0.017225, -0.329455,   0.327346,  0.527086 }, /* output layer */
  { -0.016978,  0.265793,  -0.382149, -0.532109 }
 }
};
   /* Array 'weight' contains node weights. */
   /* Indices are [layer#][node#][weight#]. */
   /* First weight for each node is the bias weight. */
   /* The other 3 are the weights on the inputs from */
   /* the previous layer. */
```

```
void main()
{
for(;;)  /* endless loop */
   {
   /* Following section assigns neural network inputs.              */
   /* The inputs used here are examples from the training data set. */
   /* These inputs would normally by the three sensor readings.     */
   redin = 143;  /* Assign red neural network input. */
      /* This must be in the range 0-255. */
   greenin  = 111;  /* Assign green neural network input. */
      /* This must be in the range 0-255. */
   nirin = 118;  /* Read near infrared (NIR) neural network input. */
      /* This must be in the range 0-255. */

   /* Following section scales inputs. */
   nodeout[0][0] = (redin-redmin)/(redmax-redmin)*2.0-1.0;
      /* scaled red input */
   nodeout[0][1] = (greenin-greenmin)/(greenmax-greenmin)*2.0-1.0;
      /* scaled green input */
   nodeout[0][2] = (nirin-nirmin)/(nirmax-nirmin)*2.0-1.0;
      /* scaled nir input */

   /* Following section passes inputs through network. */
   /* Loop through layers of neural network.           */
   for (layer=1 ; layer <= NUMLAYERS-1 ; layer=layer+1)
      {
      /* Loop through nodes of one layer. */
      for (node=0 ; node <= numnodes[layer]-1 ; node=node+1)
         {
         /* Initialize weighted sum with weighted bias. */
         sum = 1.0 * weight[layer][node][0];

         /* Following loop calculates weighted sum of inputs to   */
         /* one neural network node.  Variable i is weight index. */
         /* Loop through inputs of one node.                      */
         for (i=1 ; i <= numnodes[layer-1] ; i=i+1)
            {
            sum = sum + nodeout[layer-1][i-1] * weight[layer][node][i];
            }
         /* Calculate output of one node by using activation */
         /* function.                                        */
         nodeout[layer][node] = tanh(sum);
         }
      }

   /* Following section scales the neural network outputs. */
   output1 = (nodeout[3][0] + 0.8) / 1.6;  /* "plant" output.  */
   output2 = (nodeout[3][1] + 0.8) / 1.6;  /* "no plant" output. */

   /* Decide whether or not plant has been detected (whether or */
   /* not to turn on spray nozzle).
   if (output1 > output2)
      /* Plant detected. */
      printf("PLANT DETECTED\n");
   else
      /* No plant detected. */
      printf("NO PLANT DETECTED\n");

   }  /* end of endless FOR loop */
}  /* end of main */
```

APPENDIX D

PROGRAM LISTING OF FINAL SPRAYER UNIT SOFTWARE

```
/*  Ronnie Evans  */
/*  This program is a modified form of SPRAY17.C.                    */
/*  This program implements a history queue of past decisions on     */
/*  whether to spray or not.  All floating point operations have     */
/*  been replaced with integer operations for speed.                 */
/*  Intended for compilation by Introl C compiler and execution      */
/*  on the 68HC11 computer.                                          */
/*  Incorporates new weights obtained from training on data from     */
/*  prototype 3.                                                     */
/*  Program uses 256-location lookup table for tanh function.        */
/*  This program takes red, green, and nir sensor readings,          */
/*  runs them through a neural network model and decides whether     */
/*  or not a plant is present, and writes to PORT B accordingly      */
/*  to turn a sprayer on or off.                                     */
/*  The three inputs are assigned in 'main'.                         */
/*  The algorithm for finding the two network outputs (called        */
/*  out1 and out2 below) is the same as that used by NeuralWare      */
/*  on a forward pass (test) of the network.                         */

#include <stdio.h>
#include <math.h>
#include "h11reg.h"
#include "terminal.h"

#define DONOTHING
#define numlayers 4  /* including input layer */
#define qlength 19  /* length of history queue */
#define histsize 100  /* size of entire array that holds history */
      /* queue (nozzle state queue). */
int toggle_count, toggle_flag;
int redmin =  28;  /* min. red sensor reading in training data set. */
int redmax = 172;  /* max. red sensor reading in training data set. */
int greenmin  =  21;  /* min. green sensor reading in training  */
      /* data set. */
int greenmax  = 114;  /* max. green sensor reading in training  */
      /* data set. */
int nirmin =  15;  /* min. near infrared sensor reading in training */
      /* data set. */
int nirmax = 170;  /* max. near infrared sensor reading in training */
      /* data set. */
      /* min and max readings in training file TRAIN8.NNA. */
      /* these are used to scale the inputs. */
int layer, node, i, plant;
      /* used to pass data through neural network. */
long int redin, greenin, nirin, sum, factor1, factor2, factor3;
      /* used to pass data through neural network. */
float out1, out2;  /* neural network outputs */
int numnodes[numlayers] = {3, 3, 3, 2};
      /* numnodes array contains number of nodes in each layer,  */
      /* including input layer.  */
long int nodeout[numlayers][4];
      /* nodeout array contains node outputs. */
      /* indices are [layer#][node#]. */
```

```
long int weight[numlayers][3][4] =
{{{  0,          0,          0,          0         }   /* nothing */
 },
 {{ -12750000, 19064, 1469, -24568 },  /* hidden layer 1 */
  { 22500000, -105620, -17973, 133572 },
  { -29620000, 41628, 3470, -51873 }
 },
 {{ -11220000, 2710, -10344, 5863 },  /* hidden layer 2 */
  { 15540000, -4303, 10980, -7685 },
  { -5050000, -877, 19715, -3022 }
 },
 {{ 1720000, -3295, 3273, 5271 },  /* output layer */
  { -1700000, 2658, -3821, -5321 }
 }
};
      /* Array 'weight' contains node weights. */
      /* Indices are [layer#][node#][weight#]. */
      /* First weight for each node is the bias weight. */
      /* The other 3 are the weights on the inputs from the */
      /* previous layer. */

/* Variables for history table (nozzle state queue). */
int history[histsize];
int frontptr = 0;  /* points to front of q (where data is read) */
int rearptr = qlength; /* points to end of q (where data is written) */

/* Variables for hyperbolic tangent lookup table. */
long int minlookup = -200000000;  /* min. operand for tanh in lookup */
      /* table. */
long int maxlookup = 200000000; /* max. operand for tanh in lookup */
      /* table. */
int index;  /* index into hyperbolic lookup table. */

/* hyperbolic tangent lookup table */
static int mytanh[] =
{
-9640 ,    /*  mytanh[ 0 ]  */
-9629 ,    /*  mytanh[ 1 ]  */
-9617 ,    /*  mytanh[ 2 ]  */
-9605 ,    /*  mytanh[ 3 ]  */
-9593 ,    /*  mytanh[ 4 ]  */
-9580 ,    /*  mytanh[ 5 ]  */
-9567 ,    /*  mytanh[ 6 ]  */
-9554 ,    /*  mytanh[ 7 ]  */
-9540 ,    /*  mytanh[ 8 ]  */
-9526 ,    /*  mytanh[ 9 ]  */
-9511 ,    /*  mytanh[ 10 ]  */
-9496 ,    /*  mytanh[ 11 ]  */
-9480 ,    /*  mytanh[ 12 ]  */
-9464 ,    /*  mytanh[ 13 ]  */
-9447 ,    /*  mytanh[ 14 ]  */
-9430 ,    /*  mytanh[ 15 ]  */
-9413 ,    /*  mytanh[ 16 ]  */
-9394 ,    /*  mytanh[ 17 ]  */
-9376 ,    /*  mytanh[ 18 ]  */
-9357 ,    /*  mytanh[ 19 ]  */
-9337 ,    /*  mytanh[ 20 ]  */
-9316 ,    /*  mytanh[ 21 ]  */
-9295 ,    /*  mytanh[ 22 ]  */
-9274 ,    /*  mytanh[ 23 ]  */
-9251 ,    /*  mytanh[ 24 ]  */
```

```
-9228 ,    /*  mytanh[ 25 ]  */
-9205 ,    /*  mytanh[ 26 ]  */
-9180 ,    /*  mytanh[ 27 ]  */
-9155 ,    /*  mytanh[ 28 ]  */
-9130 ,    /*  mytanh[ 29 ]  */
-9103 ,    /*  mytanh[ 30 ]  */
-9076 ,    /*  mytanh[ 31 ]  */
-9048 ,    /*  mytanh[ 32 ]  */
-9019 ,    /*  mytanh[ 33 ]  */
-8989 ,    /*  mytanh[ 34 ]  */
-8959 ,    /*  mytanh[ 35 ]  */
-8927 ,    /*  mytanh[ 36 ]  */
-8895 ,    /*  mytanh[ 37 ]  */
-8862 ,    /*  mytanh[ 38 ]  */
-8828 ,    /*  mytanh[ 39 ]  */
-8793 ,    /*  mytanh[ 40 ]  */
-8757 ,    /*  mytanh[ 41 ]  */
-8720 ,    /*  mytanh[ 42 ]  */
-8681 ,    /*  mytanh[ 43 ]  */
-8642 ,    /*  mytanh[ 44 ]  */
-8602 ,    /*  mytanh[ 45 ]  */
-8561 ,    /*  mytanh[ 46 ]  */
-8518 ,    /*  mytanh[ 47 ]  */
-8475 ,    /*  mytanh[ 48 ]  */
-8430 ,    /*  mytanh[ 49 ]  */
-8384 ,    /*  mytanh[ 50 ]  */
-8337 ,    /*  mytanh[ 51 ]  */
-8288 ,    /*  mytanh[ 52 ]  */
-8238 ,    /*  mytanh[ 53 ]  */
-8187 ,    /*  mytanh[ 54 ]  */
-8135 ,    /*  mytanh[ 55 ]  */
-8081 ,    /*  mytanh[ 56 ]  */
-8026 ,    /*  mytanh[ 57 ]  */
-7969 ,    /*  mytanh[ 58 ]  */
-7912 ,    /*  mytanh[ 59 ]  */
-7852 ,    /*  mytanh[ 60 ]  */
-7791 ,    /*  mytanh[ 61 ]  */
-7729 ,    /*  mytanh[ 62 ]  */
-7665 ,    /*  mytanh[ 63 ]  */
-7599 ,    /*  mytanh[ 64 ]  */
-7532 ,    /*  mytanh[ 65 ]  */
-7464 ,    /*  mytanh[ 66 ]  */
-7393 ,    /*  mytanh[ 67 ]  */
-7321 ,    /*  mytanh[ 68 ]  */
-7248 ,    /*  mytanh[ 69 ]  */
-7173 ,    /*  mytanh[ 70 ]  */
-7095 ,    /*  mytanh[ 71 ]  */
-7017 ,    /*  mytanh[ 72 ]  */
-6936 ,    /*  mytanh[ 73 ]  */
-6854 ,    /*  mytanh[ 74 ]  */
-6770 ,    /*  mytanh[ 75 ]  */
-6684 ,    /*  mytanh[ 76 ]  */
-6596 ,    /*  mytanh[ 77 ]  */
-6507 ,    /*  mytanh[ 78 ]  */
-6415 ,    /*  mytanh[ 79 ]  */
-6322 ,    /*  mytanh[ 80 ]  */
-6227 ,    /*  mytanh[ 81 ]  */
-6130 ,    /*  mytanh[ 82 ]  */
-6031 ,    /*  mytanh[ 83 ]  */
-5930 ,    /*  mytanh[ 84 ]  */
-5828 ,    /*  mytanh[ 85 ]  */
```

```
-5723 ,    /*  mytanh[ 86 ]  */
-5617 ,    /*  mytanh[ 87 ]  */
-5509 ,    /*  mytanh[ 88 ]  */
-5398 ,    /*  mytanh[ 89 ]  */
-5286 ,    /*  mytanh[ 90 ]  */
-5172 ,    /*  mytanh[ 91 ]  */
-5056 ,    /*  mytanh[ 92 ]  */
-4939 ,    /*  mytanh[ 93 ]  */
-4819 ,    /*  mytanh[ 94 ]  */
-4698 ,    /*  mytanh[ 95 ]  */
-4575 ,    /*  mytanh[ 96 ]  */
-4450 ,    /*  mytanh[ 97 ]  */
-4323 ,    /*  mytanh[ 98 ]  */
-4195 ,    /*  mytanh[ 99 ]  */
-4065 ,    /*  mytanh[ 100 ]  */
-3933 ,    /*  mytanh[ 101 ]  */
-3799 ,    /*  mytanh[ 102 ]  */
-3664 ,    /*  mytanh[ 103 ]  */
-3528 ,    /*  mytanh[ 104 ]  */
-3390 ,    /*  mytanh[ 105 ]  */
-3250 ,    /*  mytanh[ 106 ]  */
-3109 ,    /*  mytanh[ 107 ]  */
-2967 ,    /*  mytanh[ 108 ]  */
-2823 ,    /*  mytanh[ 109 ]  */
-2678 ,    /*  mytanh[ 110 ]  */
-2532 ,    /*  mytanh[ 111 ]  */
-2385 ,    /*  mytanh[ 112 ]  */
-2236 ,    /*  mytanh[ 113 ]  */
-2087 ,    /*  mytanh[ 114 ]  */
-1936 ,    /*  mytanh[ 115 ]  */
-1785 ,    /*  mytanh[ 116 ]  */
-1632 ,    /*  mytanh[ 117 ]  */
-1479 ,    /*  mytanh[ 118 ]  */
-1325 ,    /*  mytanh[ 119 ]  */
-1171 ,    /*  mytanh[ 120 ]  */
-1016 ,    /*  mytanh[ 121 ]  */
-861 ,    /*  mytanh[ 122 ]  */
-705 ,    /*  mytanh[ 123 ]  */
-548 ,    /*  mytanh[ 124 ]  */
-392 ,    /*  mytanh[ 125 ]  */
-235 ,    /*  mytanh[ 126 ]  */
-78 ,    /*  mytanh[ 127 ]  */
 78 ,    /*  mytanh[ 128 ]  */
 235 ,    /*  mytanh[ 129 ]  */
 392 ,    /*  mytanh[ 130 ]  */
 548 ,    /*  mytanh[ 131 ]  */
 705 ,    /*  mytanh[ 132 ]  */
 861 ,    /*  mytanh[ 133 ]  */
 1016 ,    /*  mytanh[ 134 ]  */
 1171 ,    /*  mytanh[ 135 ]  */
 1325 ,    /*  mytanh[ 136 ]  */
 1479 ,    /*  mytanh[ 137 ]  */
 1632 ,    /*  mytanh[ 138 ]  */
 1785 ,    /*  mytanh[ 139 ]  */
 1936 ,    /*  mytanh[ 140 ]  */
 2087 ,    /*  mytanh[ 141 ]  */
 2236 ,    /*  mytanh[ 142 ]  */
 2385 ,    /*  mytanh[ 143 ]  */
 2532 ,    /*  mytanh[ 144 ]  */
 2678 ,    /*  mytanh[ 145 ]  */
 2823 ,    /*  mytanh[ 146 ]  */
```

```
2967 ,    /*  mytanh[ 147 ]  */
3109 ,    /*  mytanh[ 148 ]  */
3250 ,    /*  mytanh[ 149 ]  */
3390 ,    /*  mytanh[ 150 ]  */
3528 ,    /*  mytanh[ 151 ]  */
3664 ,    /*  mytanh[ 152 ]  */
3799 ,    /*  mytanh[ 153 ]  */
3933 ,    /*  mytanh[ 154 ]  */
4065 ,    /*  mytanh[ 155 ]  */
4195 ,    /*  mytanh[ 156 ]  */
4323 ,    /*  mytanh[ 157 ]  */
4450 ,    /*  mytanh[ 158 ]  */
4575 ,    /*  mytanh[ 159 ]  */
4698 ,    /*  mytanh[ 160 ]  */
4819 ,    /*  mytanh[ 161 ]  */
4939 ,    /*  mytanh[ 162 ]  */
5056 ,    /*  mytanh[ 163 ]  */
5172 ,    /*  mytanh[ 164 ]  */
5286 ,    /*  mytanh[ 165 ]  */
5398 ,    /*  mytanh[ 166 ]  */
5509 ,    /*  mytanh[ 167 ]  */
5617 ,    /*  mytanh[ 168 ]  */
5723 ,    /*  mytanh[ 169 ]  */
5828 ,    /*  mytanh[ 170 ]  */
5930 ,    /*  mytanh[ 171 ]  */
6031 ,    /*  mytanh[ 172 ]  */
6130 ,    /*  mytanh[ 173 ]  */
6227 ,    /*  mytanh[ 174 ]  */
6322 ,    /*  mytanh[ 175 ]  */
6415 ,    /*  mytanh[ 176 ]  */
6507 ,    /*  mytanh[ 177 ]  */
6596 ,    /*  mytanh[ 178 ]  */
6684 ,    /*  mytanh[ 179 ]  */
6770 ,    /*  mytanh[ 180 ]  */
6854 ,    /*  mytanh[ 181 ]  */
6936 ,    /*  mytanh[ 182 ]  */
7017 ,    /*  mytanh[ 183 ]  */
7095 ,    /*  mytanh[ 184 ]  */
7173 ,    /*  mytanh[ 185 ]  */
7248 ,    /*  mytanh[ 186 ]  */
7321 ,    /*  mytanh[ 187 ]  */
7393 ,    /*  mytanh[ 188 ]  */
7464 ,    /*  mytanh[ 189 ]  */
7532 ,    /*  mytanh[ 190 ]  */
7599 ,    /*  mytanh[ 191 ]  */
7665 ,    /*  mytanh[ 192 ]  */
7729 ,    /*  mytanh[ 193 ]  */
7791 ,    /*  mytanh[ 194 ]  */
7852 ,    /*  mytanh[ 195 ]  */
7912 ,    /*  mytanh[ 196 ]  */
7969 ,    /*  mytanh[ 197 ]  */
8026 ,    /*  mytanh[ 198 ]  */
8081 ,    /*  mytanh[ 199 ]  */
8135 ,    /*  mytanh[ 200 ]  */
8187 ,    /*  mytanh[ 201 ]  */
8238 ,    /*  mytanh[ 202 ]  */
8288 ,    /*  mytanh[ 203 ]  */
8337 ,    /*  mytanh[ 204 ]  */
8384 ,    /*  mytanh[ 205 ]  */
8430 ,    /*  mytanh[ 206 ]  */
8475 ,    /*  mytanh[ 207 ]  */
```

```
 8518 ,    /*  mytanh[ 208 ]  */
 8561 ,    /*  mytanh[ 209 ]  */
 8602 ,    /*  mytanh[ 210 ]  */
 8642 ,    /*  mytanh[ 211 ]  */
 8681 ,    /*  mytanh[ 212 ]  */
 8720 ,    /*  mytanh[ 213 ]  */
 8757 ,    /*  mytanh[ 214 ]  */
 8793 ,    /*  mytanh[ 215 ]  */
 8828 ,    /*  mytanh[ 216 ]  */
 8862 ,    /*  mytanh[ 217 ]  */
 8895 ,    /*  mytanh[ 218 ]  */
 8927 ,    /*  mytanh[ 219 ]  */
 8959 ,    /*  mytanh[ 220 ]  */
 8989 ,    /*  mytanh[ 221 ]  */
 9019 ,    /*  mytanh[ 222 ]  */
 9048 ,    /*  mytanh[ 223 ]  */
 9076 ,    /*  mytanh[ 224 ]  */
 9103 ,    /*  mytanh[ 225 ]  */
 9130 ,    /*  mytanh[ 226 ]  */
 9155 ,    /*  mytanh[ 227 ]  */
 9180 ,    /*  mytanh[ 228 ]  */
 9205 ,    /*  mytanh[ 229 ]  */
 9228 ,    /*  mytanh[ 230 ]  */
 9251 ,    /*  mytanh[ 231 ]  */
 9274 ,    /*  mytanh[ 232 ]  */
 9295 ,    /*  mytanh[ 233 ]  */
 9316 ,    /*  mytanh[ 234 ]  */
 9337 ,    /*  mytanh[ 235 ]  */
 9357 ,    /*  mytanh[ 236 ]  */
 9376 ,    /*  mytanh[ 237 ]  */
 9394 ,    /*  mytanh[ 238 ]  */
 9413 ,    /*  mytanh[ 239 ]  */
 9430 ,    /*  mytanh[ 240 ]  */
 9447 ,    /*  mytanh[ 241 ]  */
 9464 ,    /*  mytanh[ 242 ]  */
 9480 ,    /*  mytanh[ 243 ]  */
 9496 ,    /*  mytanh[ 244 ]  */
 9511 ,    /*  mytanh[ 245 ]  */
 9526 ,    /*  mytanh[ 246 ]  */
 9540 ,    /*  mytanh[ 247 ]  */
 9554 ,    /*  mytanh[ 248 ]  */
 9567 ,    /*  mytanh[ 249 ]  */
 9580 ,    /*  mytanh[ 250 ]  */
 9593 ,    /*  mytanh[ 251 ]  */
 9605 ,    /*  mytanh[ 252 ]  */
 9617 ,    /*  mytanh[ 253 ]  */
 9629 ,    /*  mytanh[ 254 ]  */
 9640 ,    /*  mytanh[ 255 ]  */
};
```

```
int main()
{
for(;;)  /* endless loop */
{
H11ADCTL = 0x10;  /* initiate analog-to-digital conversion. */
while(H11ADCTL != 0x90) DONOTHING;  /* wait for conversion to  */
      /* complete. */
redin = H11ADR1;  /* read red sensor section output. */
      /* Result is in the range 0-255. */
greenin = H11ADR2;  /* read green (green) sensor section output. */
      /* Result is in the range 0-255. */
nirin = H11ADR3;  /* read near infrared sensor section output. */
      /* Result is in the range 0-255. */

/* Following section scales neural network inputs. */
nodeout[0][0] = 20000*(redin-redmin)/(redmax-redmin)-10000;
      /* scaled red input */
nodeout[0][1] = 20000*(greenin-greenmin)/(greenmax-greenmin)-10000;
      /* scaled green input */
nodeout[0][2] = 20000*(nirin-nirmin)/(nirmax-nirmin)-10000;
      /* scaled nir input */

/* Following section passes inputs through neural network. */
/* Loop through layers. */
for (layer=1 ; layer < numlayers ; layer=layer+1)
   {
   /* Loop through nodes of one layer. */
   for (node=0 ; node < numnodes[layer] ; node=node+1)
      {
      /* Initialize weighted sum with weighted bias to this node. */
      sum = weight[layer][node][0];

      /* Following loop calculates weighted sum of inputs to   */
      /* one neural network node.  Variable i is weight index. */
      /* This loops through inputs of one node.                */
      for (i=1 ; i < numnodes[layer-1]+1 ; i=i+1)
         {
         /* Add weighted input to weighted sum. */
         sum = sum + nodeout[layer-1][i-1] * weight[layer][node][i];
         }
      /* Calculate index into scaled tanh(sum) lookup table. */
      /* This index is in the range 0 to 255.                */
      factor1 = (sum - minlookup) / 100;
      factor2 = (maxlookup-minlookup) / 100;
      factor3 = 255 * factor1;
      index = factor3 / factor2;

      /* If index is out of range of lookup table then use first or */
      /* last entry in table accordingly.                           */
      if (index < 0) index = 0;
      if (index > 255) index = 255;

      /* Calculate output of node by accessing tanh lookup table. */
      nodeout[layer][node] = mytanh[index];
      }  /* End of loop through nodes of a layer. */
   }  /* End of loop through layers of net. */

/* Make plant/no plant decision and put 0 or 1 at rear of */
/* history queue (nozzle state queue).                    */
if (nodeout[3][0] > nodeout[3][1]) history[rearptr] = 1;
else history[rearptr] = 0;
```

```
/* Get nozzle state (on or off) from front of nozzle state queue */
/* and update nozzle state accordingly.  The spray nozzle        */
/* controlled by the 68HC11 Port B bit 0, a digital output.      */
if (history[frontptr] == 1) H11PORTB = 1; /* turn nozzle on  */
else H11PORTB = 0; /* turn nozzle off */

/* Increment pointer to rear of queue in memory.                 */
/* If end of allotted memory reached then return to beginning. */
rearptr = rearptr + 1;
if (rearptr == histsize) rearptr = 0;

/* Increment pointer to front of queue in memory.                */
/* If end of allotted memory reached then return to beginning. */
frontptr = frontptr + 1;
if (frontptr == histsize) frontptr = 0;

}  /* end of endless for loop */
}  /* end main */
```

APPENDIX E

NEURAL NETWORK TRAINING AND TESTING DATA SETS

Format of each line of the training and testing data sets:

aaa bbb ccc d e ! fggghiij

where

aaa = Red reflected radiation sensor reading in the range 0-255.

bbb = Green reflected radiation sensor reading in the range 0-255.

ccc = Near infrared reflected radiation sensor reading in the range 0-255.

d = Target neural network output for the "plant" output.
"0" if no plant was present in the viewing area of the sensor when the line of data was recorded.
"1" if a plant was present in the viewing area of the sensor when the line of data was recorded.

e = Target neural network output for the "no plant" output.
"1" if no plant was present in the viewing area of the sensor when the line of data was recorded.
"0" if a plant was present in the viewing area of the sensor when the line of data was recorded.

! = An exclamation point indicating to the neural network development software that the remainder of the line is a comment and is to be ignored.

f = "0" if the outdoor lighting was cloudy when the line of data was recorded.
"1" if the outdoor lighting was sunny when the line of data was recorded.

ggg = Three letter code for the city in Oklahoma near the collection site for the soil on the test stand.

has Haskell	goo Goodwell	tip Tipton
alt Altus	per Perkins	sti Stillwater
man Mangum	lah Lahoma	cob Ft. Cobb
chi Chickasha		

h = "0" if the soil on the test stand was dry when the line of data was recorded.
"1" if the soil on the test stand was wet when the line of data was recorded.

ii = "00" if there was 0 percent plant cover in the viewing area when the line of data was recorded.
"20" if there was 20 percent plant cover in the viewing area when the line of data was recorded.
"50" if there was 50 percent plant cover in the viewing area when the line of data was recorded.

j = "0" if no plant (0 percent cover) was present in the viewing area when the line of data was recorded.
"1" if a plant (20 or 50 percent cover) was present in the viewing area when the line of data was recorded.

See the Neural Network Training and Testing Data Sets section for more information on this data.

NEURAL NETWORK TRAINING DATA SET

```
143 111 118   0   1  ! 1has0000
120  99 126   1   0  ! 1has0201
 99  91 148   1   0  ! 1has0501
 64  48  56   0   1  ! 1has1000
 66  60  90   1   0  ! 1has1201
 62  60 101   1   0  ! 1has1501
115  83  94   0   1  ! 1tip0000
 95  79 117   1   0  ! 1tip0201
 89  77 117   1   0  ! 1tip0501
 64  47  51   0   1  ! 1tip1000
 61  54  78   1   0  ! 1tip1201
 59  58  95   1   0  ! 1tip1501
125  87  95   0   1  ! 1per0000
111  84 105   1   0  ! 1per0201
 91  79 121   1   0  ! 1per0501
 64  44  50   0   1  ! 1per1000
 62  51  66   1   0  ! 1per1201
 61  56  86   1   0  ! 1per1501
172 114 129   0   1  ! 1man0000
143 103 140   1   0  ! 1man0201
125  94 143   1   0  ! 1man0501
111  70  84   0   1  ! 1man1000
 99  72 102   1   0  ! 1man1201
 85  67 116   1   0  ! 1man1501
142  83 110   0   1  ! 1cob0000
110  75 116   1   0  ! 1cob0201
 98  74 122   1   0  ! 1cob0501
 92  52  71   0   1  ! 1cob1000
 83  55  85   1   0  ! 1cob1201
 75  60 102   1   0  ! 1cob1501
108  78  91   0   1  ! 1has0000
 56  41  49   0   1  ! 1has1000
 98  66  82   0   1  ! 1tip0000
 60  43  51   0   1  ! 1tip1000
 98  61  81   0   1  ! 1per0000
 62  42  50   0   1  ! 1per1000
139  82 111   0   1  ! 1man0000
101  61  81   0   1  ! 1man1000
133  70 105   0   1  ! 1cob0000
 89  47  71   0   1  ! 1cob1000
 57  44  42   0   1  ! 0has0000
 47  39  40   1   0  ! 0has0201
 49  43  52   1   0  ! 0has0501
 31  25  21   0   1  ! 0has1000
 31  28  27   1   0  ! 0has1201
 32  30  36   1   0  ! 0has1501
 52  39  39   0   1  ! 0tip0000
 48  39  44   1   0  ! 0tip0201
 42  39  55   1   0  ! 0tip0501
 45  34  32   0   1  ! 0tip1000
 30  26  24   1   0  ! 0tip1201
 28  27  33   1   0  ! 0tip1501
 54  39  38   0   1  ! 0per0000
 45  36  40   1   0  ! 0per0201
 40  34  44   1   0  ! 0per0501
 36  27  22   0   1  ! 0per1000
 32  27  26   1   0  ! 0per1201
 38  35  45   1   0  ! 0per1501
 80  55  57   0   1  ! 0man0000
```

```
 73   52   59    1    0   ! 0man0201
 75   56   75    1    0   ! 0man0501
 51   35   36    0    1   ! 0man1000
 48   35   40    1    0   ! 0man1201
 46   37   52    1    0   ! 0man1501
 63   38   45    0    1   ! 0cob0000
 74   46   64    1    0   ! 0cob0201
139   86  170    1    0   ! 0cob0501
 51   31   35    0    1   ! 0cob1000
 50   34   45    1    0   ! 0cob1201
 47   36   53    1    0   ! 0cob1501
 51   41   37    0    1   ! 0has0000
 48   37   39    0    1   ! 0has1000
 80   58   62    0    1   ! 0tip0000
 40   32   26    0    1   ! 0tip1000
 64   44   45    0    1   ! 0per0000
 35   26   22    0    1   ! 0per1000
 84   54   63    0    1   ! 0man0000
 67   44   49    0    1   ! 0man1000
 69   41   48    0    1   ! 0cob0000
 49   30   34    0    1   ! 0cob1000
```

NEURAL NETWORK TESTING DATA SET

```
113  86  84   0   1  ! 1goo0000
 80  67  80   1   0  ! 1goo0201
 84  78 113   1   0  ! 1goo0501
 66  48  49   0   1  ! 1goo1000
 61  54  70   1   0  ! 1goo1201
 62  57  82   1   0  ! 1goo1501
 85  64  62   0   1  ! 1alt0000
 74  63  76   1   0  ! 1alt0201
 72  65  84   1   0  ! 1alt0501
 52  39  36   0   1  ! 1alt1000
 52  47  57   1   0  ! 1alt1201
 55  52  69   1   0  ! 1alt1501
 94  62  75   0   1  ! 1sti0000
 81  62  87   1   0  ! 1sti0201
 71  63 101   1   0  ! 1sti0501
 57  38  45   0   1  ! 1sti1000
 54  46  68   1   0  ! 1sti1201
 53  50  79   1   0  ! 1sti1501
 90  61  70   0   1  ! 1lah0000
 76  60  89   1   0  ! 1lah0201
 73  62  97   1   0  ! 1lah0501
 52  36  40   0   1  ! 1lah1000
 52  42  55   1   0  ! 1lah1201
 51  47  71   1   0  ! 1lah1501
 95  56  77   0   1  ! 1chi0000
 74  54  84   1   0  ! 1chi0201
 72  57  90   1   0  ! 1chi0501
 52  34  40   0   1  ! 1chi1000
 52  43  61   1   0  ! 1chi1201
 52  45  67   1   0  ! 1chi1501
 94  64  72   0   1  ! 1goo0000
 61  44  46   0   1  ! 1goo1000
 67  46  50   0   1  ! 1alt0000
 46  34  33   0   1  ! 1alt1000
 77  47  62   0   1  ! 1sti0000
 54  34  43   0   1  ! 1sti1000
 77  47  60   0   1  ! 1lah0000
 47  31  36   0   1  ! 1lah1000
 78  41  62   0   1  ! 1chi0000
 49  30  37   0   1  ! 1chi1000
 58  45  39   0   1  ! 0goo0000
 51  42  44   1   0  ! 0goo0201
 43  41  55   1   0  ! 0goo0501
 36  28  22   0   1  ! 0goo1000
 36  31  30   1   0  ! 0goo1201
 38  37  48   1   0  ! 0goo1501
 36  28  22   0   1  ! 0alt0000
 46  37  39   1   0  ! 0alt0201
 46  39  54   1   0  ! 0alt0501
 35  27  21   0   1  ! 0alt0000
 30  27  24   1   0  ! 0alt1201
 29  28  34   1   0  ! 0alt1501
 52  35  36   0   1  ! 0sti0000
 50  37  44   1   0  ! 0sti0201
 42  36  54   1   0  ! 0sti0501
 37  26  24   0   1  ! 0sti1000
 39  30  32   1   0  ! 0sti1201
 33  31  41   1   0  ! 0sti1501
 63  41  44   0   1  ! 0lah0000
```

```
43  33  37   1   0  ! 0lah0201
40  34  46   1   0  ! 0lah0501
59  37  44   0   1  ! 0lah1000
36  29  31   1   0  ! 0lah1201
35  34  46   1   0  ! 0lah1501
45  29  31   0   1  ! 0chi0000
41  29  35   1   0  ! 0chi0201
35  29  42   1   0  ! 0chi0501
30  21  18   0   1  ! 0chi1000
30  24  24   1   0  ! 0chi1201
32  29  39   1   0  ! 0chi1501
51  39  34   0   1  ! 0goo0000
39  30  25   0   1  ! 0goo1000
44  33  28   0   1  ! 0alt0000
29  23  15   0   1  ! 0alt1000
55  38  39   0   1  ! 0sti0000
43  30  30   0   1  ! 0sti1000
60  40  41   0   1  ! 0lah0000
45  31  31   0   1  ! 0lah1000
68  40  49   0   1  ! 0chi0000
36  25  22   0   1  ! 0chi1000
```

APPENDIX F

PHOTODIODE DATA SHEETS

HAMAMATSU

Photodiodes

Including Si, GaAsP and GaP Photodiodes

HAMAMATSU

HAMAMATSU PHOTONICS K.K., Solid State Division
1126, Ichino-cho, Hamamatsu City, 435 Japan
Telephone: 0534/34-3311, Fax: 0534/35-1037, Telex: 4225-185

Main Products

Silicon Photodiodes
PIN Silicon Photodiodes
Silicon Avalanche Photodiodes
GaAsP Photodiodes
PCD Linear Image Sensors
Position-Sensitive Detectors
Phototransistors
Infrared Detectors
CdS Photoconductive Cells
Optoisolators
Opto-Hybrid IC
Infrared LED
Pulsed Laser Diodes

Hamamatsu also supplies:
Photoelectric Tubes
Imaging Tubes
Specialty Lapms
Measuring Video Systems

Sales Offices

ASIA:
HAMAMATSU PHOTONICS K.K.

325-6, Sunayama-cho,
Hamamatsu City, 430 Japan
Telephone: 0534/52-2141, Fax: 0534/56-7889,
Telex: 4225-186

U.S.A.:
HAMAMATSU CORPORATION

Main Office
360 Foothill Road, P.O. BOx 6910,
Bridgewater, N.J. 08807-0910, U.S.A.
Telephone: 201/231-0960, Fax: 201/231-1539

Western U.S.A. Office
2444 Moorpark Avenue, Suite 312
San Jose, Calif. 95128, U.S.A.
Telephone: 408/292-8603, Fax: 408/279-1886

United Kingdom:
HAKUTO INTERNATIONAL (UK) LTD.
Eleanor House, 33-35 Eleanor Cross Road,
Waltham Cross, Hertfordshire, EN8 7LF Engalnd
Telephone: 0992-769090, Fax: 0992-763300
Telex: 299288

France, Spain, Portugal, Belgium:
HAMAMATSU PHOTONICS FRANCE
49/51, Rue de la Vanne,
92120 Montrouge, France
Telephone: (1) 46 55 47 58, Fax: (1) 46 55 36 65
Telex: 631-895

W. Germany:
HAMAMATSU PHOTONICS DEUTSCHLAND GmbH
Arzbergerstr. 10,
D-8036 Herrsching am Ammersee,
West Germany
Telephone: 08152-375-0, Fax: 08152-2658
Telex: 527731

Sweden, Norway, Finland:
LAMBDA ELECTRONICS AB
Grevgatan 39, S-114 53,
Stockholm, Sweden
Telephone: 08-662 06 10, Fax: 08-663 40 26
Telex: 13952

Denmark:
LAMBDA ELECTRONICS AS
Nilivej 8c,
DK-2000 Frederiksberg F, Denmark
Telephone: 01-19 15 55, Fax: 01-19 37 59

Italy
HESA S.P.A.
Viale Teodorico 19/1, 20149 Milano, Italy
Telephone: (02)317-551, Fax: (02)341-384
Telex: 331219

Hong Kong:
S&T ENTERPRISES LTD.
Room 404, Block B,
Watson's Estate, Watson Road,
North Point, Hong Kong
Telephone: 5-784921, Fax: 580-73126
Telex: 73942

Taiwan, R.O.C.:
S&T ENTERPRISES LTD.
Taiwan Branch
No. 75, Section 4, Nanking East Road,
Taipei, Taiwan
Telephone: 02-715-3403, Fax: 02-712-9240
Telex: 22590

KORYO ELECTRONICS CO., LTD.
Min-Seng Trade Bldg.,
No. 342, Min-Seng East Road,
Taipei, Taiwan,
Telephone: 02-505-6470, Fax: 02-501-1552
Telex: 25335

Korea:
SANGSOO SANGSA CO.
Suite 421, Sunmyunghoi Bldg.,
24-2, Yoido-Dong, Youngdeungpo-ku,
Seoul, Korea
Telephone: 02-782-8514, Fax: 02-784-6062
Telex: 22565

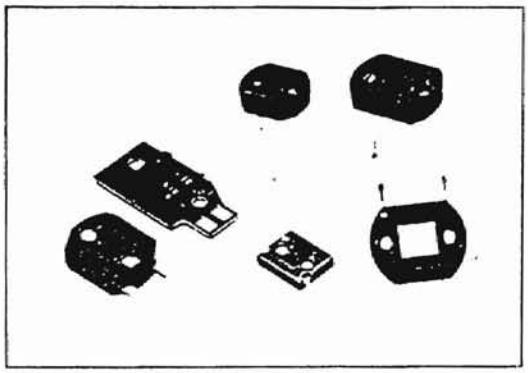
Singapore:
S&T ENTERPRISES LTD.
Singapore Branch
80, Genting Lane,
Unit 03-02, 03-05, Genting Block
Ruby Industrial Complex
Singapore 1334.
Telephone: 74594235, Fax: 065-7469630
Telex: 24784

Quality, technology, and service are part of every product.

DEC/87
Supersedes JAN/85
CR-6500 Printed in Japan

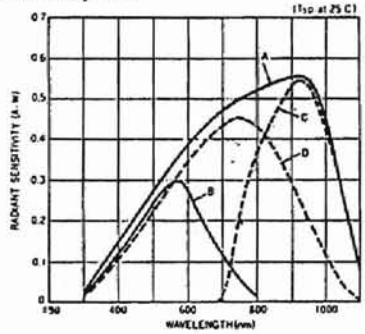
Silicon Photodiodes (Visible Light/Visible Light to IR Photometry)

Type No.	Features	Ⓐ Outlines / Window Materials	Photosensitive surface		Spectral Response		Characteristics (25°C)			
							Typical Radiant Sensitivity (A/W)			
			Size (mm)	Effective Area (mm^2)	Range / Mark (nm)	Peak Wavelength (nm)	Peak Wavelength	560nm GaP LED	633nm He-Ne Laser	930nm GaAs LED
S1087-01	For visible light to near IR	㉒/R	1.3 x 1.3	1.6	300~1100/A	940±50	0.55	0.35	0.42	0.55
S1087	For visible light	㉓/V			320~730/B	560±20	0.3	0.3	0.16	–
S1087-03	For visible light, high-speed response									
S2164	Chip carrier type, for visible light to near IR	㉓/R	1.3 x 1.3	1.6	300~1100/A	940±50	0.55	0.35	0.42	0.55
S2164-01	Chip carrier type, visible light cutoff type	㉒/F			700~1100/C	940±50	0.55	–	–	0.55
S2357	Glass epoxy substrate, for visible light	㉚/V	1.1 x 1.7	1.78	320~730/B	560±20	0.3	0.3	0.16	–
S1133-01	For visible light to near IR	㉔/R	2.4 x 2.8	6.6	300~1100/A	940±50	0.55	0.35	0.42	0.55
S1133-11	For visible light to near IR, high speed response									
S1133	For visible light	㉕/V			320~730/B —— 320~840/B	560±20	0.3	0.3	0.2	–
S1133-03	For visible light, high-speed response									
S1133-02	For visible light, high sensitivity									
S1133-12	For visible light, high sensitivity, high speed									
S1133-14	For visible light to near IR, high speed, low IR sensitivity	㉔/R			300~1060/D	740±50	0.45	0.3	0.4	0.25
S1133-05	For visible light, high-speed response, low IR ratio	㉕/V			320~730/B	560±20	0.3	0.3	0.2	–
S1787-08	For visible light to near IR, SIP case	㉘/R	2.4 x 2.8	6.6	300~1100/A	940±50	0.55	0.35	0.42	0.55
S1787-04	For visible light, SIP case	㉘/V			320~730/B —— 320~840/B	560±20	0.3	0.3	0.2	–
S1787-06	For visible light, high-speed response, SIP case									
S1787-05	For visible light, high sensitivity, SIP case									
S1787-07	For visible light, high speed, high sensitivity, SIP case									

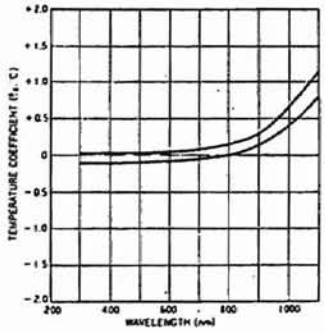


Ⓐ See pages 34 to 37 for outlines.
Window materials are
R: Resin coating
V: Visible compensating filter
F: Visible light cutoff filter

• Spectral Response

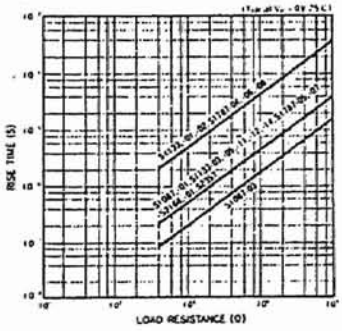


• Temperature Characteristic of I_{sh}

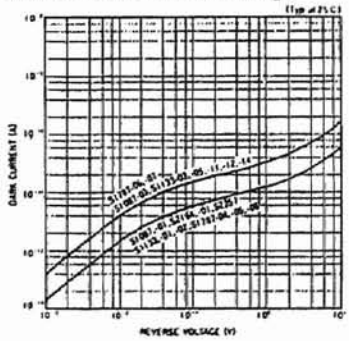


Characteristics (at 25°C)									Absolute Maximum Ratings			Type No.
Infrared Sensitivity Ratio Typ. (%)	Short Circuit Current I_{sh} 100 lux Typ. (μA)	Temperature Dependence of Short Circuit Current Typ. (%/°C)	Dark Current Id V_R = 1V Max. (pA)	Temperature Dependence of Dark Current Typ. (Times/°C)	Shunt Resistance, R_{sh} V_R = 10mV Min. (GΩ)	Shunt Resistance, R_{sh} V_R = 10mV Typ. (GΩ)	Junction Capacitance C_j V_R = 0V Typ. (pF)	Rise Time t_r V_R = 0V R_L = 1kΩ Typ. (μs)	Reverse Voltage V_{R}max (V)	Temperature Range Operating (°C)	Temperature Range Storage (°C)	
–	1.5	+0.1	10	1.12	4	30	200	0.5	10	−10~+60	−20~+70	S1087-01
10	0.13	−0.01	10	1.12	4	30	200	0.5	10	−10~+60	−20~+70	S1087
		−0.02	20		2	10	60	0.2				S1087-03
–	1.5	+0.1	10	1.12	4	30	200	0.5	10	−10~+60	−20~+70	S2164
	1.0											S2164-01
10	0.17	−0.01	10	1.12	4	30	250	0.5	10	−10~+60	−20~+70	S2357
–	5.5	+0.1	10	1.12	4	30	700	2.5	10	−10~+60	−20~+70	S1133-01
			20		2	10	200	0.5				S1133-11
10	0.54	−0.01	10	1.12	4	30	700	2.5	10	−10~+60	−20~+70	S1133
	0.50		20		2	10	200	0.5				S1133-03
20	0.85	−0.02	10	1.12	4	30	700	2.5	10	−10~+60	−20~+70	S1133-02
	0.78	−0.01	20	1.12	2	10	200	0.5	10	−10~+60	−20~+70	S1133-12
–	3.2	+0.1	20	1.12	2	10	200	0.5	10	−10~+60	−20~+70	S1133-14
8	0.50	−0.02	20	1.12	2	10	200	0.5	10	−10~+60	−20~+70	S1133-05
–	5.5	+0.1	10	1.12	4	30	700	2.5	10	−10~+60	−20~+70	S1787-08
10	0.54	−0.01	10	1.12	4	30	700	2.5	10	−10~+60	−20~+70	S1787-04
	0.50	−0.02	20	1.12	2	10	200	0.5	10	−10~+60	−20~+70	S1787-06
20	0.85	−0.01	10	1.12	4	30	700	2.5	10	−10~+60	−20~+70	S1787-05
	0.78	−0.02	20	1.12	2	10	200	0.5	10	−10~+60	−20~+70	S1787-07

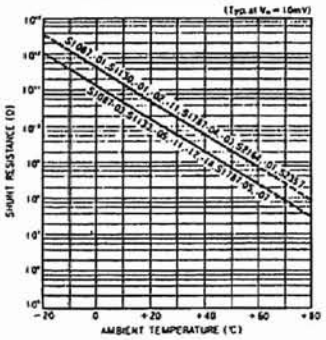
- Response Time vs. Load Resistance



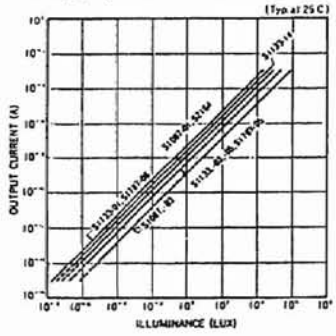
- Dark Current vs. Reverse Voltage



- Shunt Resistance vs. Temperature

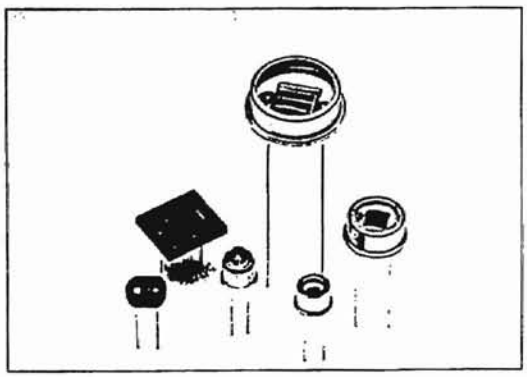


- Linearity (Representative Example)



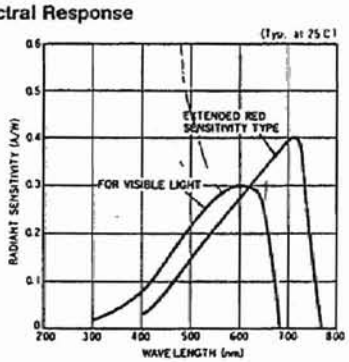
GaAsP Photodiodes

Type No.	Outlines / Window Materials Ⓐ	Package (mm)	Photosensitive Surface: Size (mm)	Photosensitive Surface: Effective Area (mm^2)	Spectral Response: Range (nm)	Spectral Response: Peak Wavelength (nm)	Characteristics (25°C): Typical Radiant Sensitivity (A/W): Peak Wavelength	Typical Radiant Sensitivity (A/W): 560nm GaP LED	Typical Radiant Sensitivity (A/W): 633nm He-Ne Laser	Short Circuit Current I_{sh}, 100 lux: Min (μA)	Short Circuit Current I_{sh}, 100 lux: Typ. (μA)
Diffusion Type (for Visible Light)											
G1115	❶/K	TO-18	1.3 x 1.3	1.66	300~680	610±30	0.3	0.29	0.29	0.12	0.15
G1116	❺/K	TO-5	2.7 x 2.7	7.26						0.45	0.6
G1117	❿/K	TO-8	5.6 x 5.6	29.3						2	2.5
G1118	㉒/R	5 x 6	1.3 x 1.3	1.66						0.12	0.15
G1120	⓳/R	8.9 x 10.1	5.6 x 5.6	29.3						2	2.5
G1122	❷/L	TO-18	1.3 x 1.3	1.66						0.4	0.5
Diffusion Type (Extended Red Sensitivity Type)											
G1735	❶/K	TO-18	1.3 x 1.3	1.66	400~760	710±30	0.4	0.22	0.29	0.2	0.25
G1736	❺/K	TO-5	2.7 x 2.7	7.26						0.8	1.1
G1737	❿/K	TO-8	5.6 x 5.6	29.3						3.5	4.5
G1738	㉒/R	5 x 6	1.3 x 1.3	1.66						0.2	0.25
G1740	⓳/R	8.9 x 10.1	5.6 x 5.6	29.3						3.5	4.5
G1742	❷/L	TO-18	1.3 x 1.3	1.66						0.7	0.8

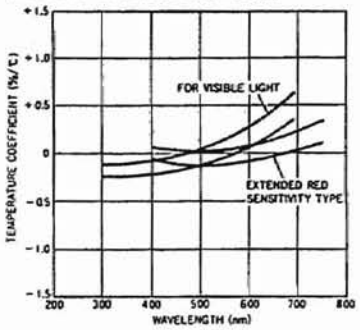


Ⓐ See pages 34 to 37 for outlines.
Window materials are
K: Borosilicate glass
L: Lens type borosilicate glass
R: Resin coating

• Spectral Response

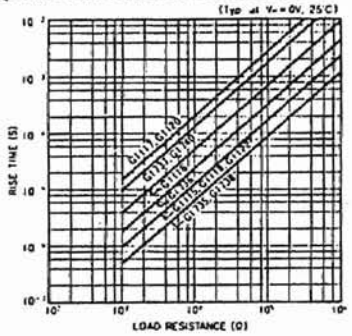


• Temperature Characteristic of I_{sh}

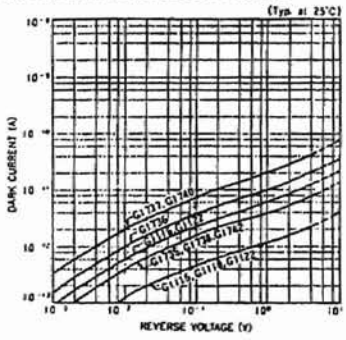


Characteristics (25°C)									Absolute Maximum Ratings			Type No.
Dark Current, Id Max.		Temperature Dependence of Dark Current Typ.	Shunt Resistance Rsh V_R = 10mV		Junction Capacitance C_t V_R = 0V Typ.	Rise Time t_r V_R = 0V R_L = 1k Ω Typ.	NEP Typ.	D* Typ.	Reverse Voltage V_{RMAX}	Temperature Range		
V_R = 10mV (pA)	V_R = 1V (pA)	(Times/°C)	Min. (GΩ)	Typ. (GΩ)	(pF)	(μs)	(W/Hz$^{1/2}$)	(cm·Hz$^{1/2}$/W)	(V)	Operating (°C)	Storage (°C)	
1	10	1.07	10	80	300	1	9×10^{-16}	1×10^{14}	5	−10~+60	−20~+80	G1115
2.5	25		4	30	1400	4	2×10^{-15}					G1116
5	50		2	15	6000	15	3×10^{-15}					G1117
1	10		10	80	300	1	9×10^{-16}				−20~+70	G1118
5	50		2	15	6000	15	3×10^{-15}					G1120
1	10		10	80	300	1	9×10^{-16}				−20~+80	G1122
2	20	1.07	5	25	250	0.5	2×10^{-15}	6×10^{13}	5	−10~+60	−20~+80	G1735
5	50		2	10	1200	1.8	3×10^{-15}					G1736
10	100		1	5	4500	10	7×10^{-15}					G1737
2	20		5	25	250	0.5	2×10^{-15}				−20~+70	G1738
10	100		1	5	4500	10	7×10^{-15}					G1740
2	20		5	25	250	0.5	2×10^{-15}				−20~+80	G1742

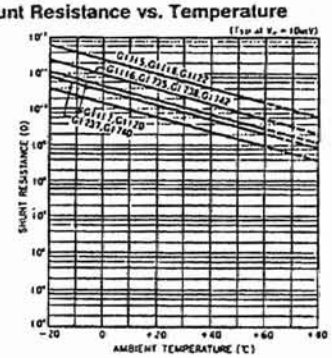
- Response Time vs. Load Resistance



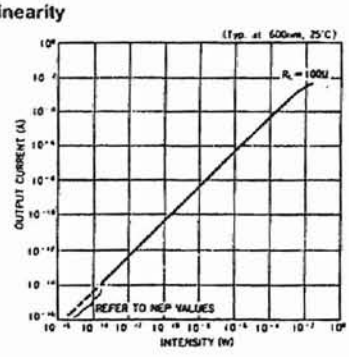
- Dark Current vs. Reverse Voltage



- Shunt Resistance vs. Temperature



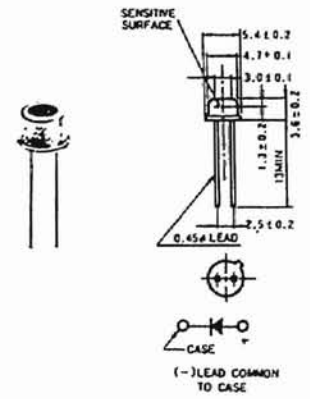
- Linearity



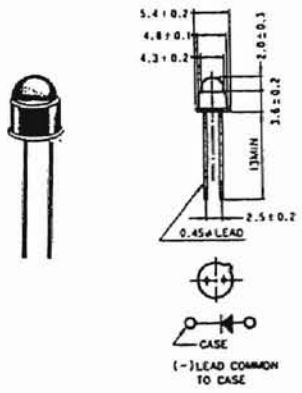
Dimensional Outlines

Unit: mm

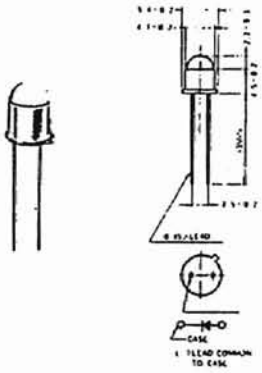
❶ S1226-18BQ etc.



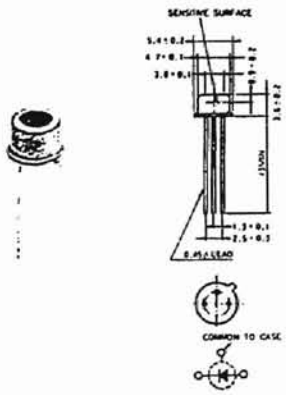
❷ S2386-18L etc.



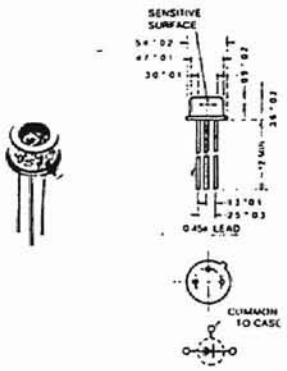
❸ S1190-01



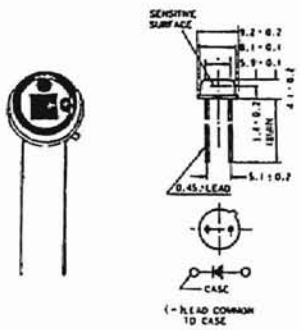
❹ S1188-02, S2216-01, etc.



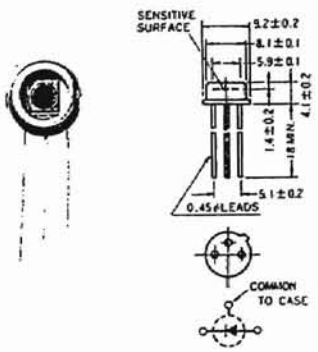
❺ S2381, S2382, S2383



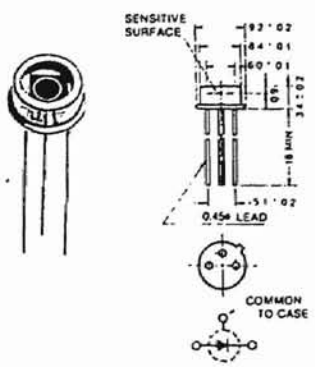
❻ S1226-5BQ etc.



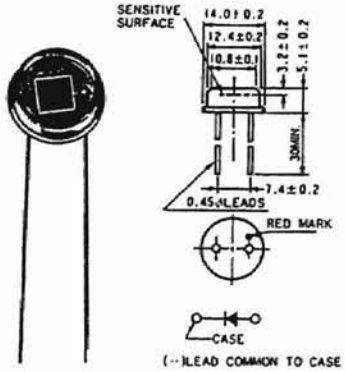
❼ S1721



❽ S2384

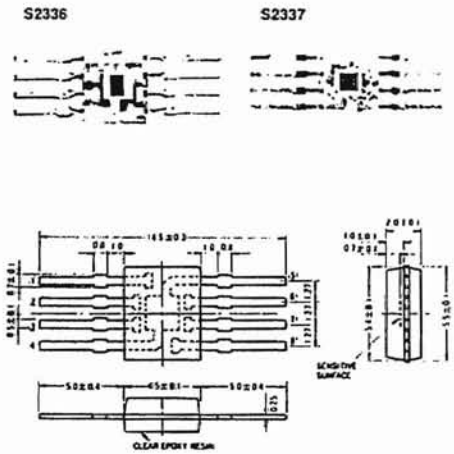


❾ S1226-8BQ etc.

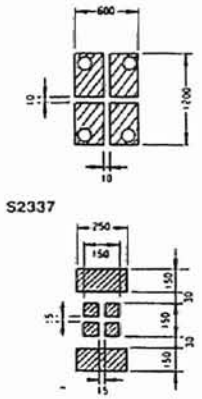


Unit: mm

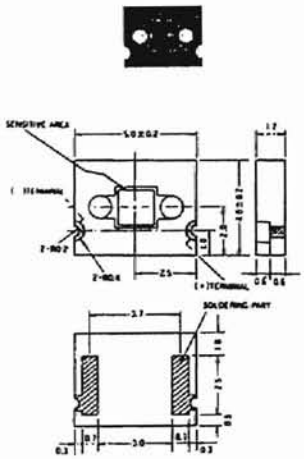
㉘ S2336, S2337



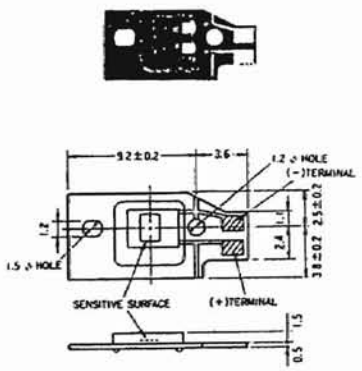
Details of Sensitive Area
(Unit: μm)



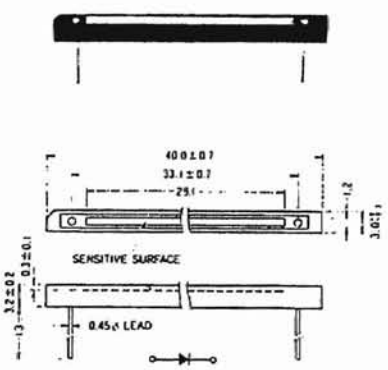
㉙ S2164, S2164-01



㉚ S2357



㉛ S2551



* The spacings of the leads in the figures are indicated as center-to-center dimensions. The photograph shows a typical type.

APPENDIX G

OPTICAL FILTER DATA SHEETS

SPINDLER & HOYER

COLORED GLASS FILTERS

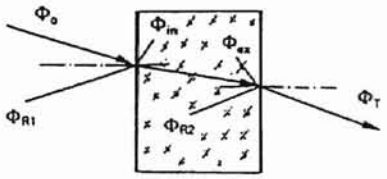
Glass filters are designed to attenuate the radiation from polychromatic sources. They either attenuate this radiation uniformly (neutral density filters) or selectively at different wavelengths (color filters). When incident light Φ_0 passes through a colored glass filter, a fraction of the energy Φ_R is reflected at both of the air/glass boundaries. Another fraction Φ_A is absorbed and the remainder Φ_T is transmitted.

The transmission factor τ is defined as the ratio of the transmitted intensity to the incident intensity and consequently characterizes the transmission behavior of a filter at a certain wavelength λ.

You will find the internal transmission factor τ_i in the filter graph. It indicates the ratio of the radiant flux within the filter.

Transmission factors τ can be calculated by multiplying τ_i by the reflection factor P (for the d-line 587,6 nm). Reflection factor P is stated for the individual filters.

The internal transmission factor τ_i allows you to determine the dependence of the filter transmission on thickness. It is plotted as an ordinate based on a $1-\log(\log\frac{1}{\tau})$ scale. This exaggerates the curves in the lower transmission region.



$$\Phi_0 = \Phi_R + \Phi_A + \Phi_T$$

$$\Phi_R = \Phi_{R1} + \Phi_{R2}$$

$$\tau = \frac{\Phi_T}{\Phi_0}$$

$$\tau_i = \frac{\Phi_{ex}}{\Phi_{en}}$$

$$\tau = \tau_i \cdot P$$

$$\tau_{i2}(\lambda) = \tau_{i1}(\lambda)^{\left(\frac{d_2}{d_1}\right)}$$

Φ_0 = incident radiation flux
Φ_R = reflected radiation flux
Φ_A = absorbed radiation flux
Φ_T = transmitted radiation flux
Φ_{ex} = inside exit face incident flux
Φ_{en} = inside entrance face incident flux
τ = transmission factor
τ_i = internal transmission factor
P = reflection factor

SPINDLER & HOYER

COLORED GLASS FILTERS SHARP CUT

	Typ	Edge position nm	Reflection Factor P	Thick-ness (mm)	Unmounted Ø 22,4 mm Part No.	Unmounted Ø 50 mm Part No.	In Mount C Aperture Ø 21,4 mm Part No.	In Mount 50 Aperture Ø 48 mm Part No.
UV-absorption filter, colorless	GG 385	385	0,905	2	37 0106	37 0014	06 3438	03 1875
Yellow filter	GG 475	475	0,915	3	37 0088	37 0013	06 3452	03 1869
Orange filter	OG 530	530	0,915	3	37 0089	37 0012	06 3453	03 1870
Red filter, bright red	OG 590	590	0,915	2	37 0090	37 0058	06 3439	03 1880
Red filter, medium red	RG 610	610	0,915	2	37 0107	37 0041	06 3440	03 1871
Red filter, dark red	RG 630	630	0,915	3	37 0081	37 0011	06 3081	03 1872
IR-filter	RG 780	780	0,915	3	37 0091	37 0019	06 3454	03 1874
IR-filter	RG 830	830	0,915	3	37 0092	37 0063	06 3455	03 1881
IR-filter	RG 850	850	0,910	3	37 0108	37 0109	06 3441	03 1884
IR selective filter for near IR	RG 9	735	0,915	2	37 0093	37 0042	06 3456	03 1873

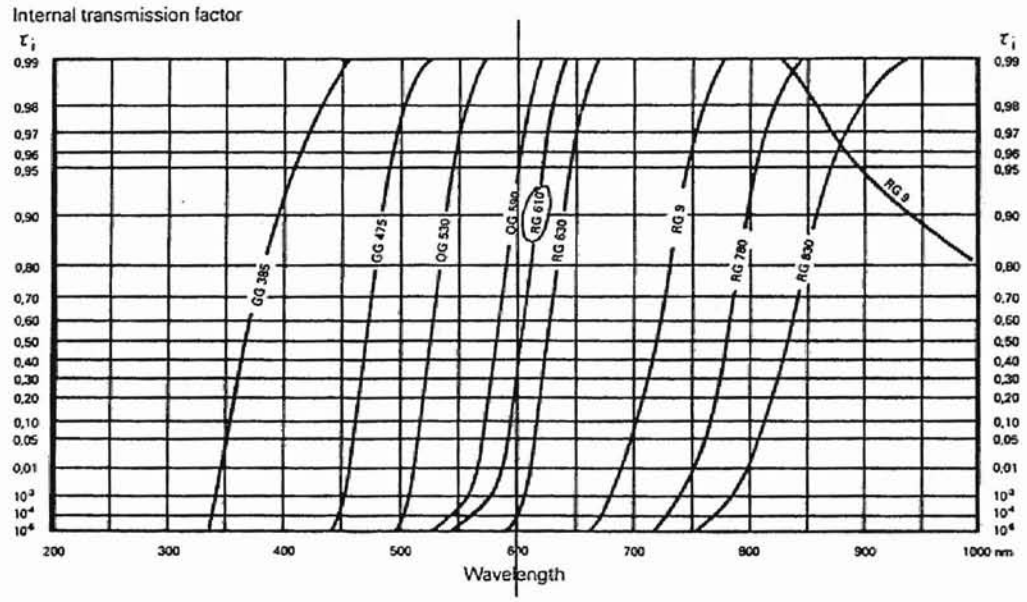
TOLERANCE

Ø 22,4 mm: −0,3 mm

Ø 50 mm: −0,6 mm

Thickness: ±0,2 mm

Bubbles:	1/5×0,25
Schlieren:	2/03
Surface quality:	5/5×0,25
Parallelism:	15'



SPINDLER & HOYER

COLORED GLASS FILTERS

	Type	max. Transmission nm	Reflection Factor P	Thickness (mm)	Unmounted Ø 22,4 mm Part No.	Unmounted Ø 50 mm Part No.	In Mount C Aperture Ø 21,4 mm Part No.	In Mount 50 Aperture Ø 48 mm Part No.
Blue-green filter moderate red transmission max. IR transmission at 2400 nm	BG 38	480	0,915	3	37 0086	37 0043	06 3450	03 1867
Blue-green filter hardly any red transmission max. IR transmission at 2400 nm	BG 18	500	0,91	5	37 0087	37 0036	06 3451	03 1866
Green filter, dark green very little red transmission max. IR transmission at 2600 nm	VG 9	520	0,91	2	37 0082	37 0015	06 3082	03 1868
Green filter, bright green considerable red transmission max. IR transmission between 1600 and 2800 nm	VG 5	530	0,91	2	37 0103	37 0055	06 3435	03 1859

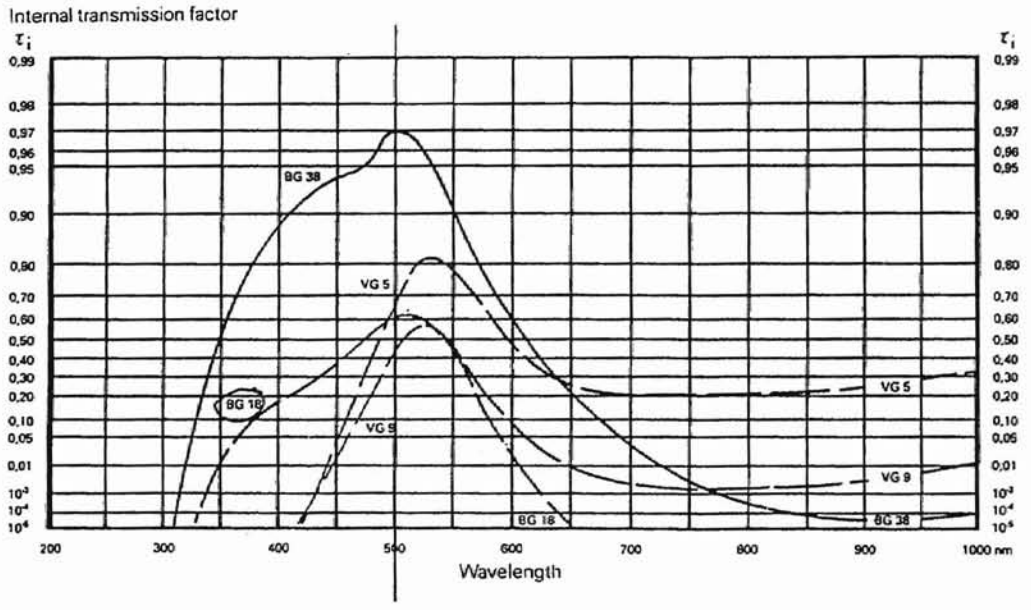
TOLERANCE

Ø 22,4 mm: −0,3 mm

Ø 50 mm: −0,6 mm

Thickness: ±0,2 mm

Bubbles:	1/5×0,25
Schlieren:	2/03
Surface quality:	5/5×0,25
Parallelism:	15'



APPENDIX H

SELECTED SECTIONS FROM CNAT (SPRAYER UNIT COMPUTER) USER'S MANUAL

APPENDIX H CONTENTS

CNAT Overview

The Chrysler Network Analysis Tool (CNAT) is a small single board controller intended to aid in the development and analysis of multiplex networks using the Chrysler developed CCD interface.

The CNAT board has the following features:

- MC68HC11A0 microprocessor
- 32K bytes of EPROM
- 32K bytes of Non-Volatile RAM
- CCD bus interface IC
- CCD bus physical layer interface with choice of terminations
- I/O connectors with port pins available for user attachment
- A "Power-On" LED
- A "CCD Transmit" LED
- A "CCD Receive" LED
- RS-232 communication at 19,200 baud to the host PC
- Single 12 volt input with on-board voltage regulation

The CNAT can be used in two different ways:

- It can operate as a stand-alone board connected to the CCD bus while executing a program stored in on-board non-volatile RAM memory. This is the 'Stand-Alone' mode.
- It can operate as an intelligent interface board connected to a CCD bus and communicate via an RS-232 serial link to an IBM compatible personal computer (PC) host. This method of operation uses communication and display software created for use on an IBM PC compatible computer. This is the 'PC' mode.

The CNAT is intended for use in a laboratory, bench top environment only. Although the CNAT was designed with protection circuitry against the automotive environment, operation in automotive enviromental or electrical conditions has not been tested and is not guarantied.

See Figure 1 for a block diagram of the CNAT board.

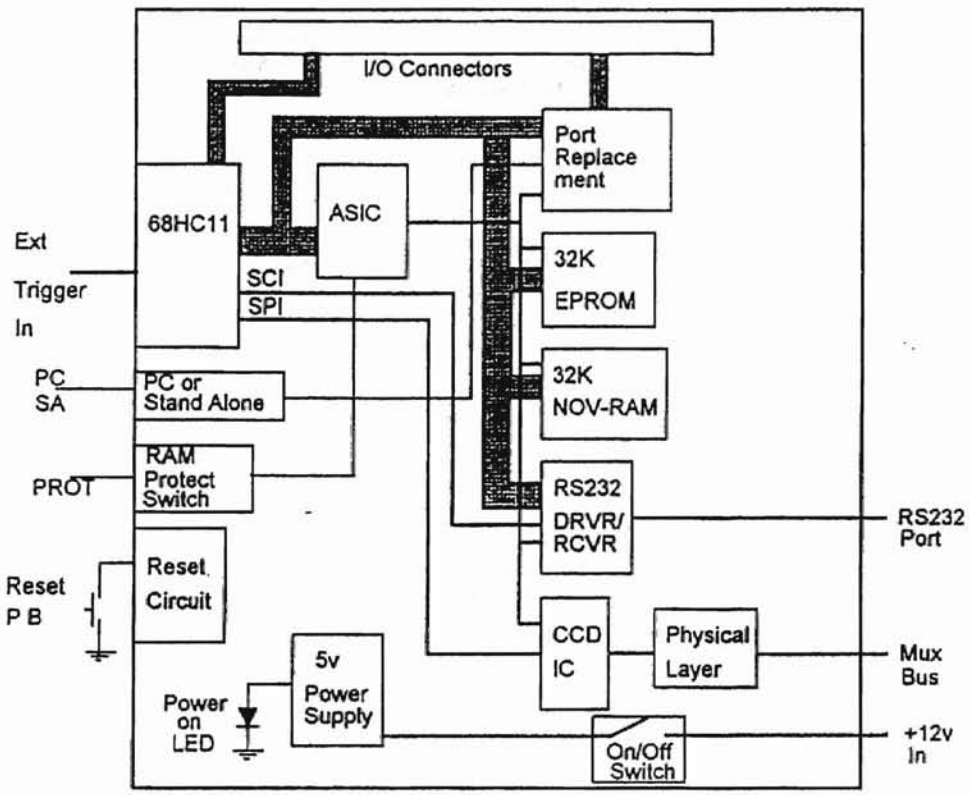


Figure 1 CNAT Block Diagram

Getting Started

To start using the CNAT you must first install the user interface software, and also set up the hardware configuration of the CNAT board itself.

Software Installation

The installation of the interface software for the CNAT is very simple. The software is not copy-protected. The interface software may be run from a floppy disk or from a hard disk.

The following example will create a sub directory on the hard disk, and copy all of the CNAT files to the sub directory. If your hard disk has a drive designation other than **C:**, or a floppy disk designation other than **A:**, then substitute that drive letter in the example.

```
C:>          mkdir c:\cnat↵

C:>          cd c:\cnat↵

C:\CNAT      xcopy a:*.* /s /v↵
```

To execute the interface program on the PC, change to the sub directory and type the program name-cnat followed by a space, then the number of the COM port the CNAT board is connected to:

```
C:\CNAT>     cnat 1↵
```

CNAT 1 uses the COM1 port and CNAT 2 uses the COM2 port. The CNAT program is easy to use and will provide on-screen help. A detailed discussion of the software is provided later in this manual.

Hardware Setup

There are a number of jumpers and switches on the CNAT board that are used to configure the board for your intended use. There are also 4 connectors to attach the board to outside signals. The individual jumpers should be set before power is applied to the board. We recommend that power to the board be turned off (via the Power ON or OFF Toggle Switch) before changes are made to jumper settings.

Here is a quick overview of the hardware jumpers, switches and connectors, and what they do.

See Figure 2 to locate the placement of these jumpers, switches and connectors.

Power ON or OFF Toggle Switch

There is a toggle switch on the board, labeled "ON" and "OFF". Leave it "OFF" until the board is configured, and the power connector (same as the MUX bus connector) is attached. Switch it to "ON" to make the board work.

Power ON LED

When the switch is in the "ON" position and there is power connected to the board via the Mux Bus Connector the LED next to the Power toggle switch will light.

Reset Pushbutton

There is a pushbutton switch on the board labeled "RESET". Push it to reset the entire board. This has the same effect as a "power-on-reset".

PC or Stand-Alone Toggle Switch

There is a toggle switch on the board, labeled "PC" and "SA" (Stand-Alone). This toggle switch determines what mode the board will operate in.

When set to "PC" it will expect to communicate with another device via the serial port upon power-on. It will continue to attempt to communicate on the serial connection, and will not execute any user program stored in non-volatile RAM until the correct command is sent via the serial interface.

When set to "SA", the board will attempt to execute a user program stored in non-volatile memory on the board. If there is not a valid program stored then the results are unpredictable.

Non-Volatile RAM Write Protect Toggle Switch

There is a toggle switch on the board, labeled "WE" (Write Enable) and "PROT" (Protect). When set to "WE" the NOV-RAM contents may be changed by the micro. When set to "PROT" writing to the NOV-RAM is prevented.

CCD Mux Bus Termination Selection Jumpers

There are three jumpers on the board that control the termination resistors for the physical layer of the CCD bus.

1. Jumper JP6, when inserted, connects a 120Ω resister between the BUS+ and BUS- terminals.
2. Jumper JP5, when inserted, connects a 13KΩ resistor from the BUS- terminal to Vcc on the board.
3. Jumper JP7, when inserted, connects a 13KΩ resistor from the BUS+ terminal to Ground on the board.

Mux Bus and Power In Connector

There is a 4 pin mini-DIN connector on the CNAT board. This connector is the connection to the CCD bus, and it is also the power connector for the board. Make sure the Power ON or OFF toggle switch is in the OFF position before attaching this connector.

Bus Active LEDs

There are two LEDs at the edge of the board, next to the Trigger Input connector. One is labeled "XMIT", for transmit, and one is labeled "RCV", for receive. These LEDs are used to indicate when the microprocessor is processing bus messages into or out of the board. They are driven by port pins from the microprocessor.

Trigger Input Connector

There is a BNC connector on the CNAT board. This connector is used to input a trigger signal to the microprocessor. This trigger input capability is provided for use with customer-written application software. The basic CBIU software and the PC CBIU software do not use the trigger input.

Serial I/O Connector

There is a 9-pin 'D' type female connector on the CNAT board. This is the connector for the RS-232 connection to the PC.

Port Expansion Connectors

There are two 40 pin female header connectors on the board. These connectors bring out the microprocessor's port and control signals for expansion of the CNAT board. Refer to the electrical schematics, supplied separately, for the details of these connections.

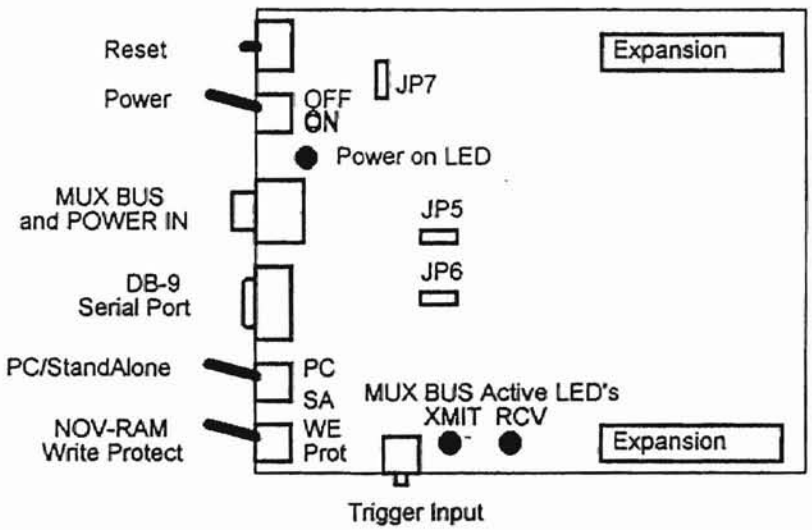


Figure 2 CNAT Board Layout

Mux Bus and Power Cable

You must provide a cable to attach the power inputs (+12 volts and ground) and the Mux Bus connections to the CNAT board. The cable should be wired as shown in Figure 3. The end of the cable that will attach to the CNAT board must have a 4-pin mini DIN type male connector on it. One of these connectors is provided with the CNAT board. The other end of this cable should have a connector that will attach properly to your source of power and to your Mux Bus signals.

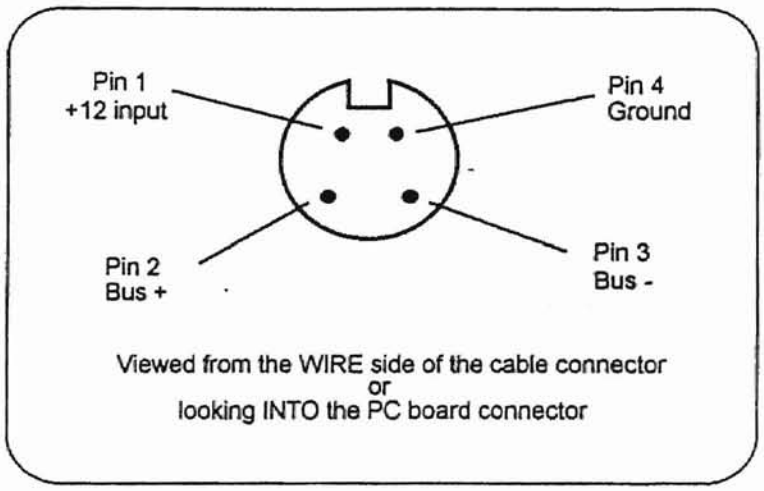


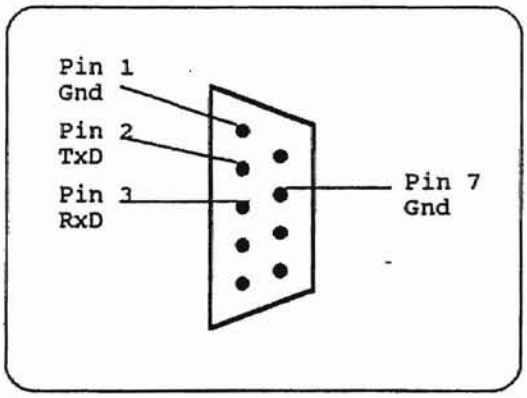
Figure 3 Mux Bus and Power Cable Pinout

Serial Cable

You must provide a cable to attach the CNAT board to the Computer. The cable should be wired as shown in Figure 4. The end of the cable that will attach to the CNAT board must have a 9-pin 'D' type male connector on it. The other end should have a connector that will attach properly to your PC's COM port connector.

RS232 Communications

The CNAT is designed to run at a baud rate of 19,200 with 8 data bits, no parity, and 1 stop bit. No other baud rates or formats are supported.



- Shell is not grounded

Figure 4 Serial Cable Pinout

Stand Alone Mode

The 68HC11 microprocessor in the CNAT module performs like any other 68HC11. A user program can be downloaded into the 68HC11's NOV-RAM. These instructions are put into the 68HC11's NOV-RAM memory as described.

The following memory maps are given to aide the user in writing executable code. The program can be executed in the stand alone mode only and is accomplished by setting the PC/SA switch to the SA position and depressing the RESET button.

CNAT Board Memory MAP

* Memory map:

PRU:	1000H - 1FFFH.
NOVRAM:	2000H - 7FFFH.
EPROM:	8000H - FFFFH.

The following information is to aid the user in writing executable code to download into the CNAT's HC11 non-volatile RAM:

-There are no useable subroutines within the CNAT CBIU software, i.e. transmit or receive.
-Microsoft and C compilers offer useable functions to implement standard routines
-The data rate is set at 19.2k baud.

Interrupt Vector Assignments for CNAT

Vector Address	Interrupt Source
7FC1 - 7FC3	STANDALONE_SCI
7FC4 - 7FC6	STANDALONE_SPI
7FC7 - 7FC9	STANDALONE_PAIE
7FCA - 7FCC	STANDALONE_PAOF
7FCD - 7FCF	STANDALONE_TOI
7FD0 - 7FD2	STANDALONE_OC5I
7FD3 - 7FD5	STANDALONE_OC4I
7FD6 - 7FD8	STANDALONE_OC3I
7FD9 - 7FDB	STANDALONE_OC2I
7FDC - 7FDE	STANDALONE_OC1I
7FDF - 7FE1	STANDALONE_IC3I
7FE2 - 7FE4	STANDALONE_IC2I
7FE5 - 7FE7	STANDALONE_IC1I
7FE8 - 7FEA	STANDALONE_RTI
7FEB - 7FED	STANDALONE_IRQ
7FEE - 7FF0	STANDALONE_XIRQ
7FF1 - 7FF3	STANDALONE_SWI
7FF4 - 7FF6	STANDALONE_IOPC
7FF7 - 7FF9	STANDALONE_COPF
7FFA - 7FFC	STANDALONE_COPMF
7FFD - 7FFF	STANDALONE_RESET

Note: Each vector must contain a jump or a branch to the interrupt service routine.

Specifications

Environmental

Storage temperature: -40 to 85 degrees C.

Operational temperature: 0 to 70 degrees C.

Electrical

Maximum supply current, 300 milliamps.

Input voltage (From P2.1 to P2.4):

- Line transient for Vcc <= 6V: 60V for t ≤ 120 milliseconds.
- Reverse voltage duration: Infinite for V ≥ -100V.
- Minimum operational input voltage: 6.25V.
- Maximum input voltage: 26V.

* I/O port electrical specifications:

Valid logic input low voltage: ≤ 1V (with respect to P2.4).

Valid logic input high voltage: ≥ 3.5V (with respect to P2.4).

Valid logic output low voltage: ≤ 0.4V (with respect to P2.4).
(Load = 1.6 milliamps)

Valid logic output high voltage: ≥ 4.2V (with respect to P2.4).
(Load = -0.8 milliamps)

Analog voltage range: 0 to 5V.
(JP8: 19, 21, 23, 25, 27, 29, 31, 33 with respect to P2.4).

Minimum pulse width low duration for interrupt acknowledge,
(JP8: 1, 3) 1020 nsec (IRQ & XIRQ)

Minimum pulse width low or high for valid input capture,
(JP8: 3, 5, 7, 9, 11, 13, 15, 17) 1020 nsec (PA0 - PA2)

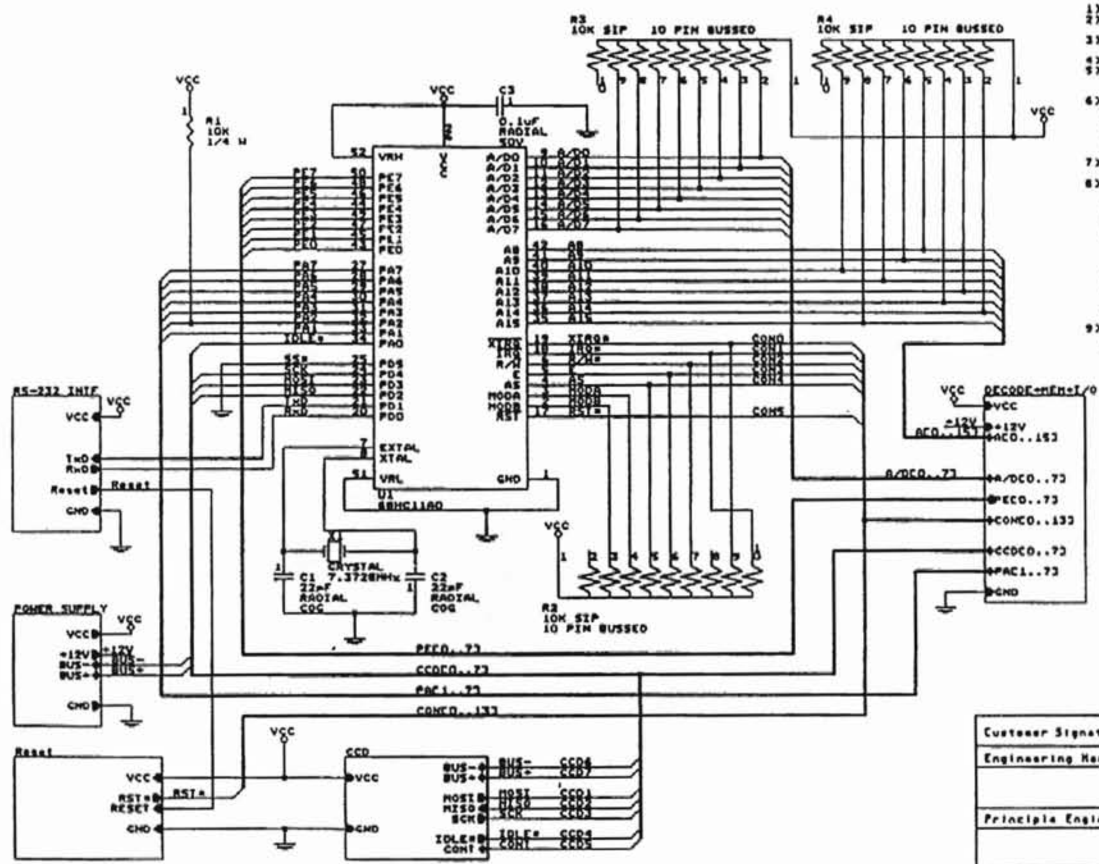
CCD interface:

Biasing pull-up and pull down resistor values: 13K ohms ± %5.

Network terminating resistor values: 120 ohms ± %5.

Transmission data rate: 7812.5Hz ± <1Hz.

Microprocessor crystal frequency: 4.9152MHz ± <492Hz.



Notes:

1) P.C.Board Size 7" X 4.5".
2) 72 Way Connector near the 68HC24 (PRU).
3) Heat Sink on the LM2940 to be 1.45"X1.0"X7.9".
4) LM2940 to be laid down.
5) Total supply current to the piggy-back board to be less than 400 mA.
6) Support holes for the piggy-board referenced to pin 1 of the 72 Way connector to be at (0.40,0.05), (-3.90,0.05), (-3.90,-3.45) (0.40,-3.45).
7) Include Ground clip 0.062 hole.
8) Silkscreen to contain the following:
   a) component outlines where possible.
   b) component ref designators.
   c) pin numbers for the connectors at least every 10 pins.
   d) "The Dearborn Group, Inc."
   e) Date code.
   f) 0.7 x 0.3 solid block for serial number.
   g) Labels for the switches and LED's.
9) Realitive location of 40 Way connectors must remain the same.

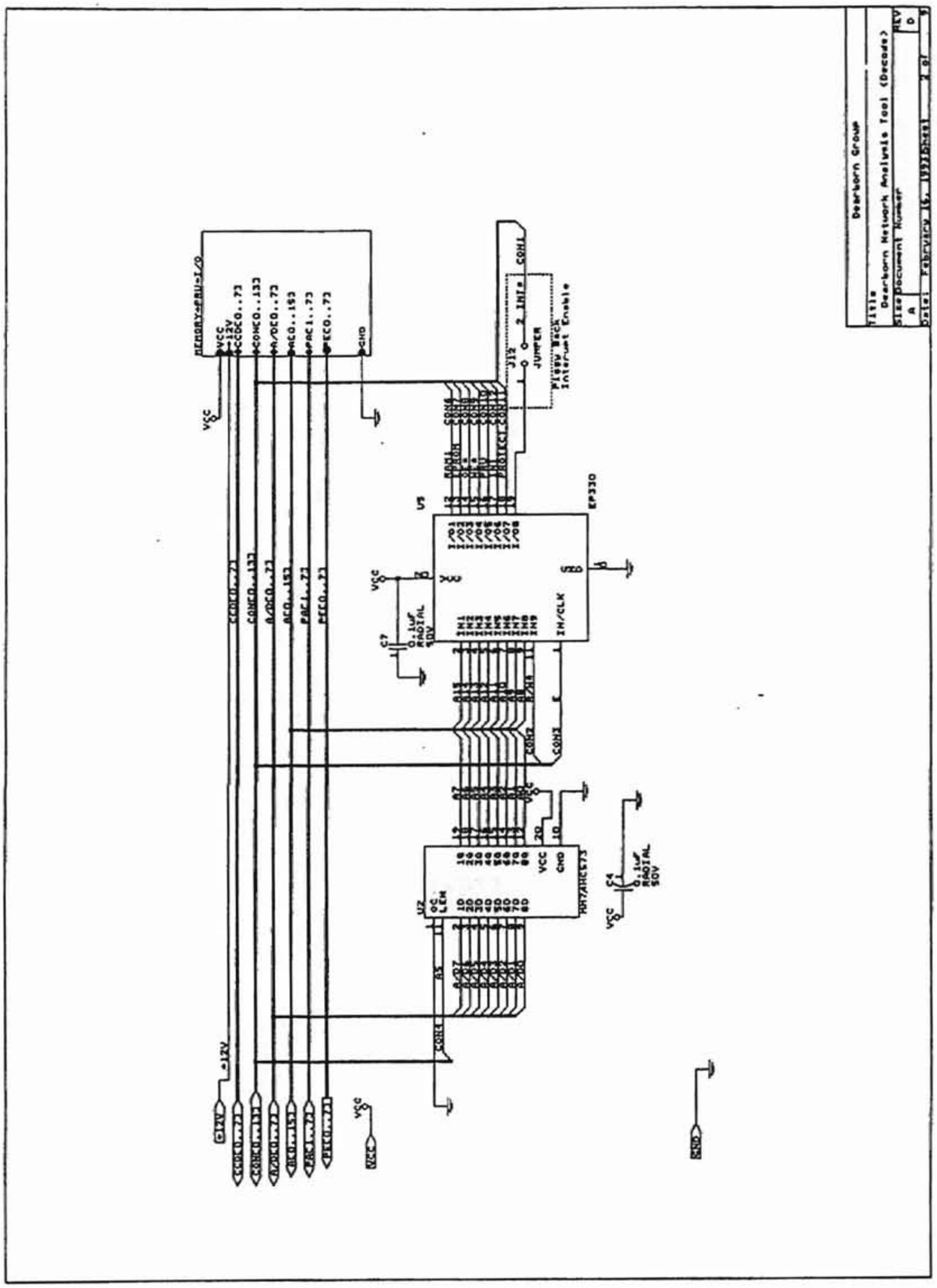
Customer Signatures	Date
Engineering Manager:	
Principle Engineer:	
Project Engineer:	

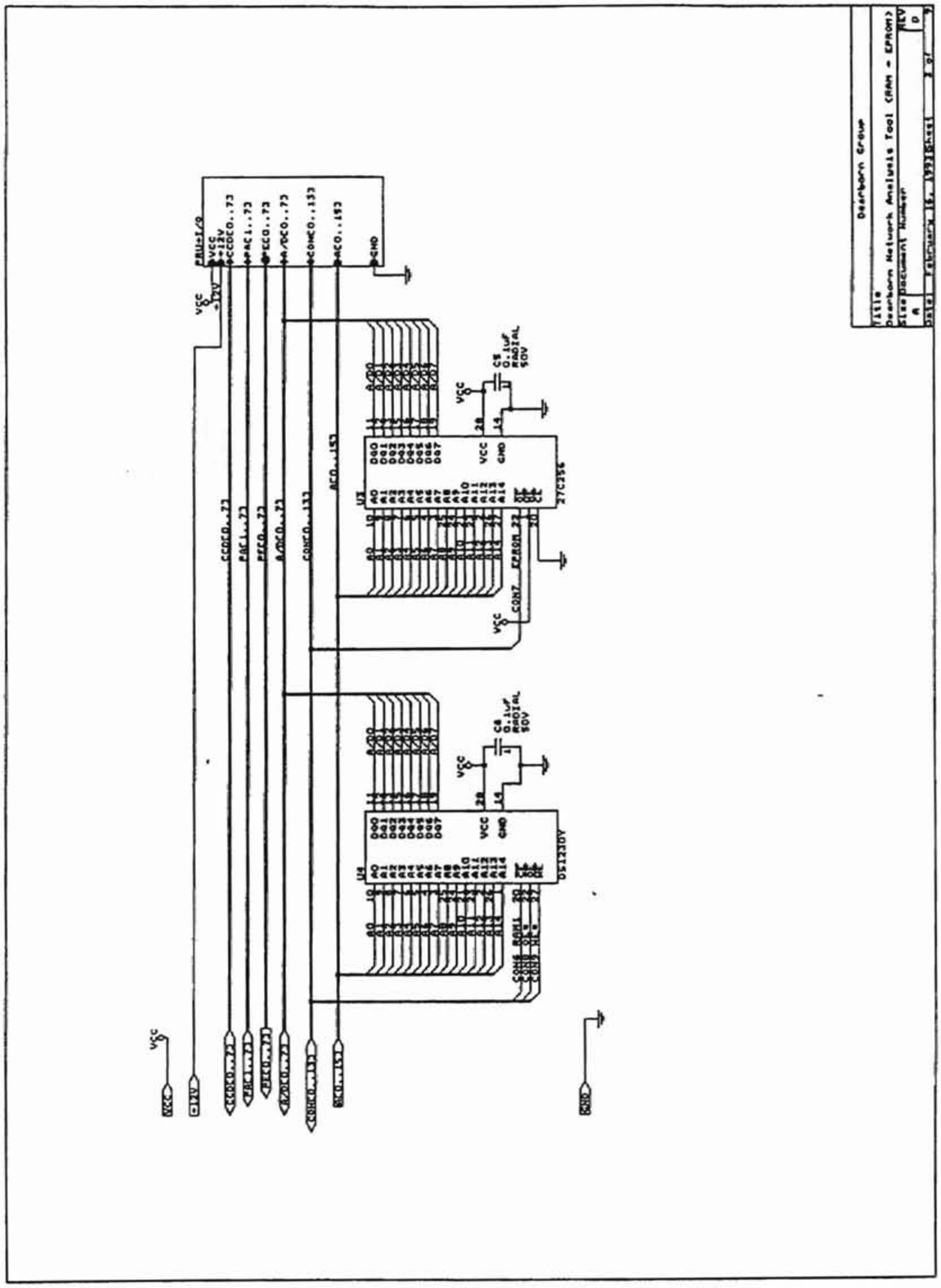
Dearborn Group

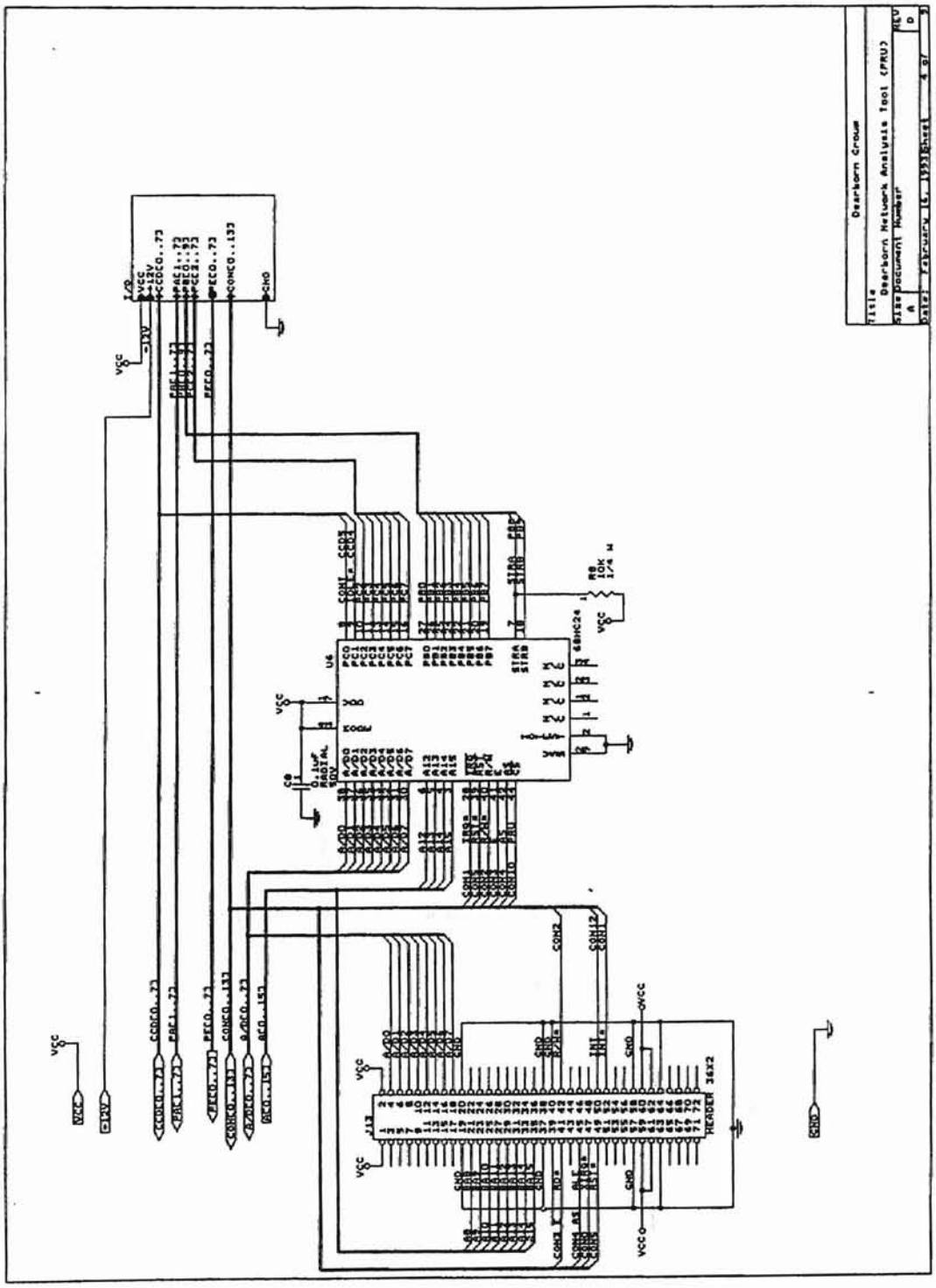
Title: Dearborn Network Analysis Tool (MICRO)

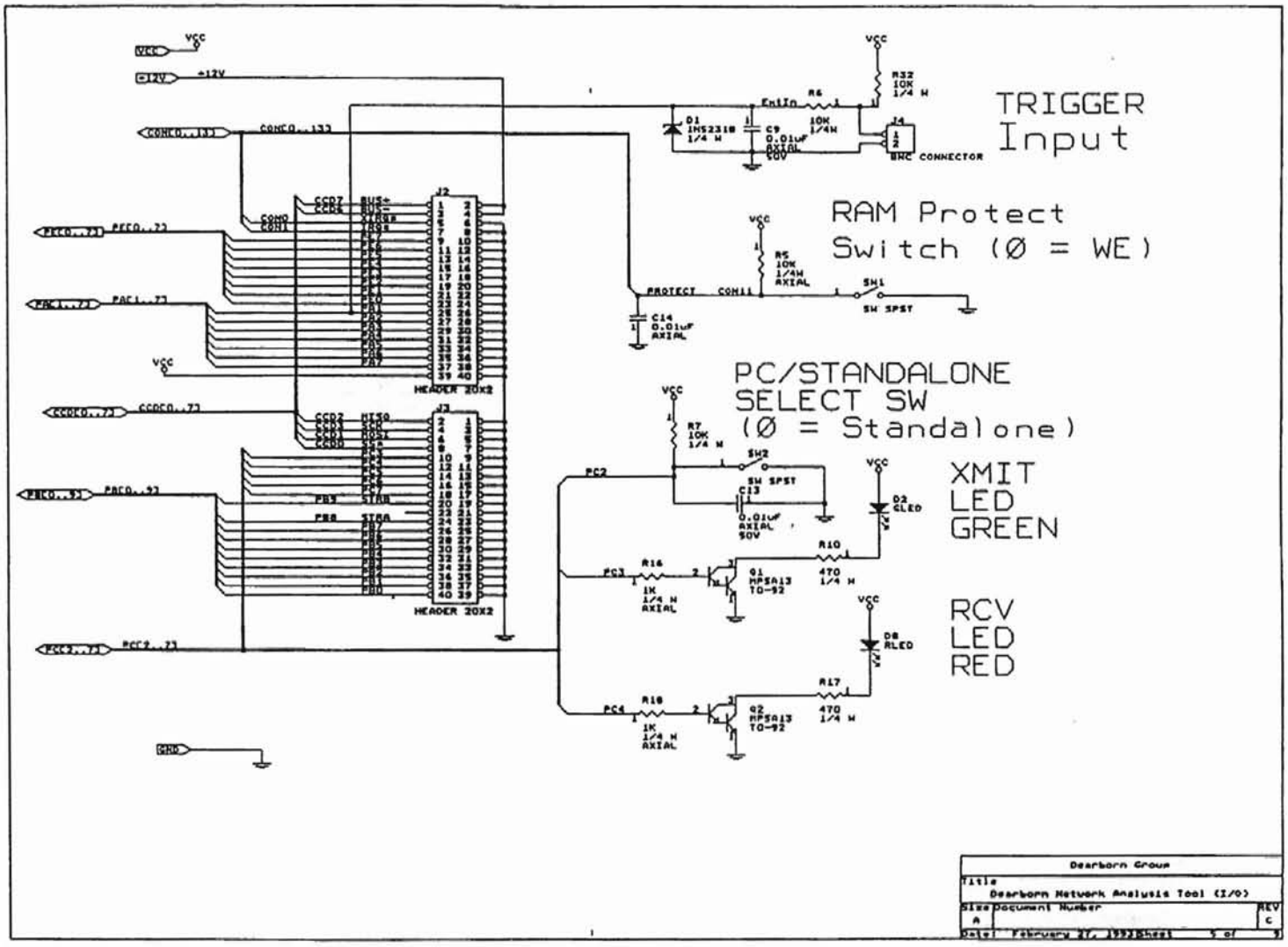
Size: A | Document Number: | REV: C

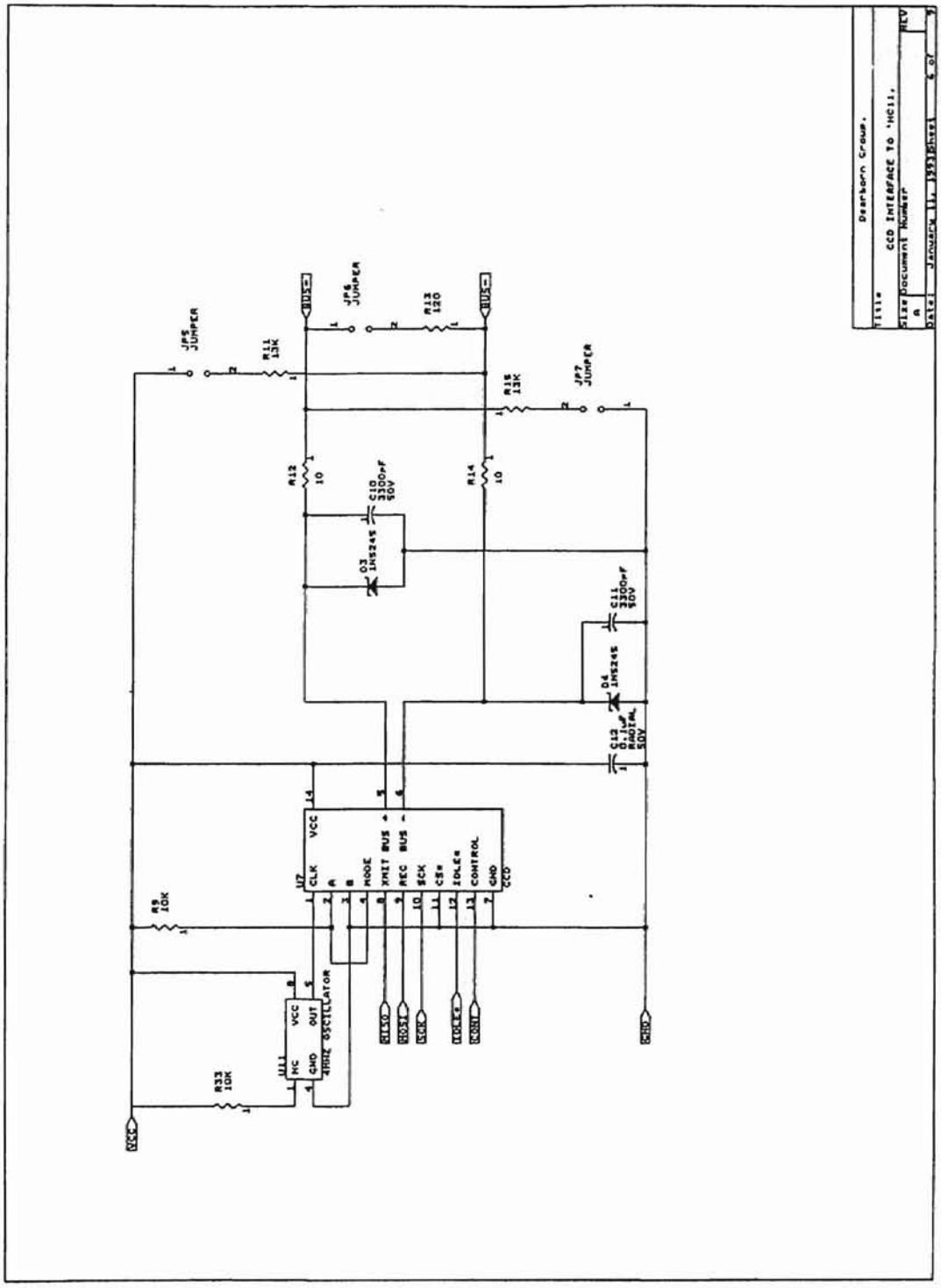
Date: February 16, 1993 | Sheet 1 of 9

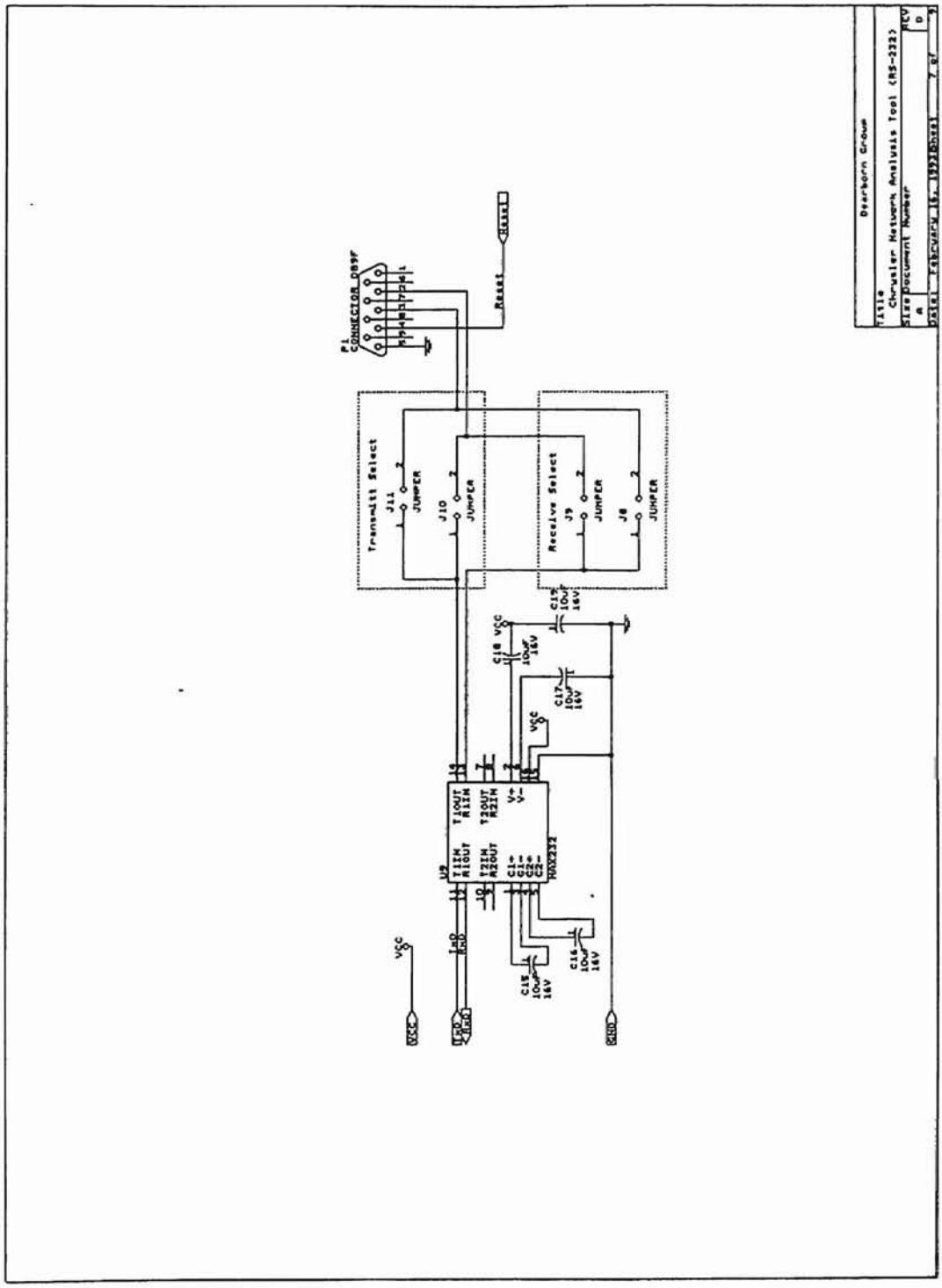

Dearborn Group
Title
Dearborn Network Analysis Tool (Decode)
Size
Document Number
A
REV
D
Date:
Sheet
2 of
MEMORY+PRU+I/O
VCC
+12V
CON1
CON2
CON3
CON4
U5
EP330
U2
JUMPER
0.1uF
RADIAL
50V
GND

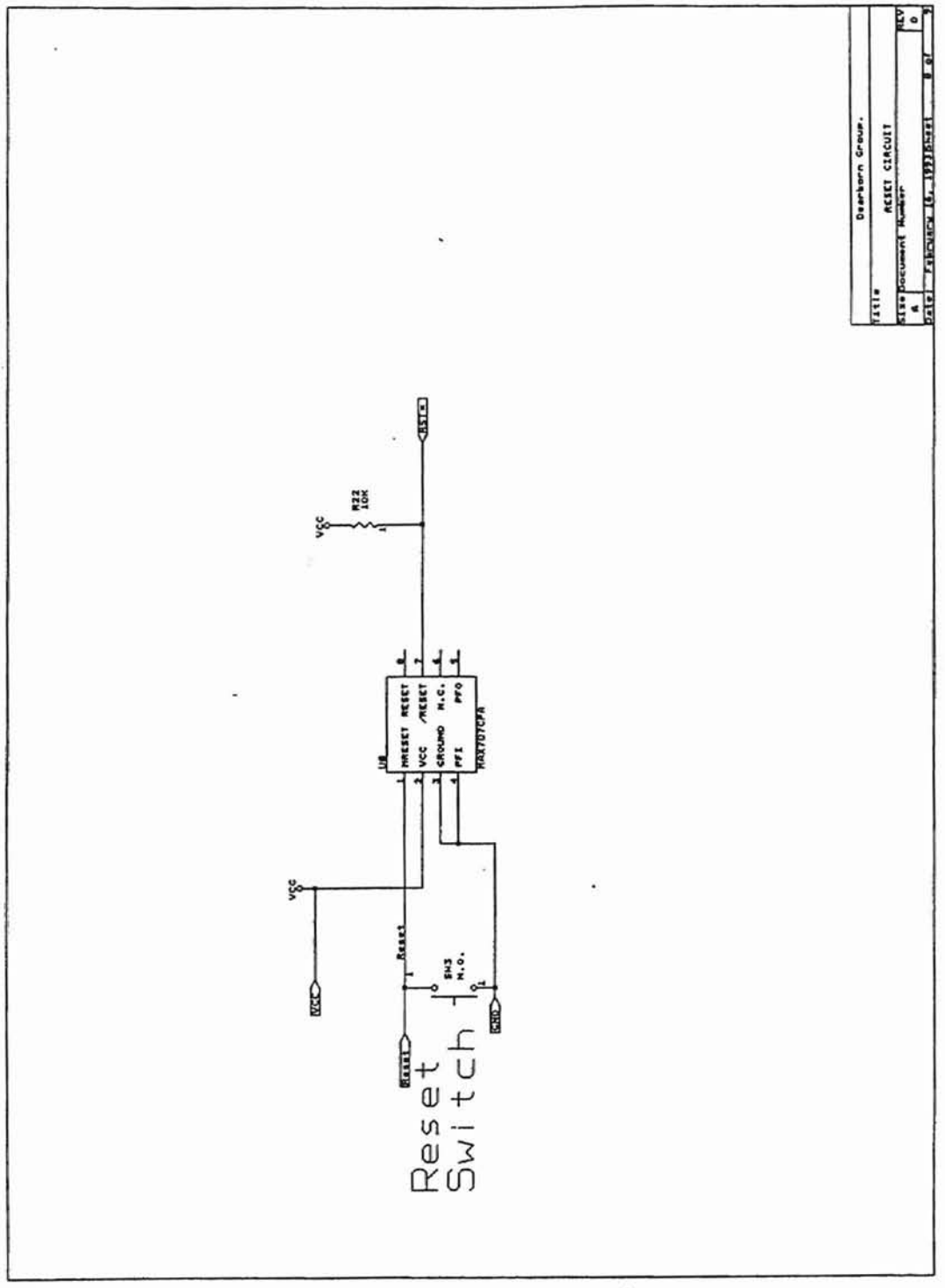


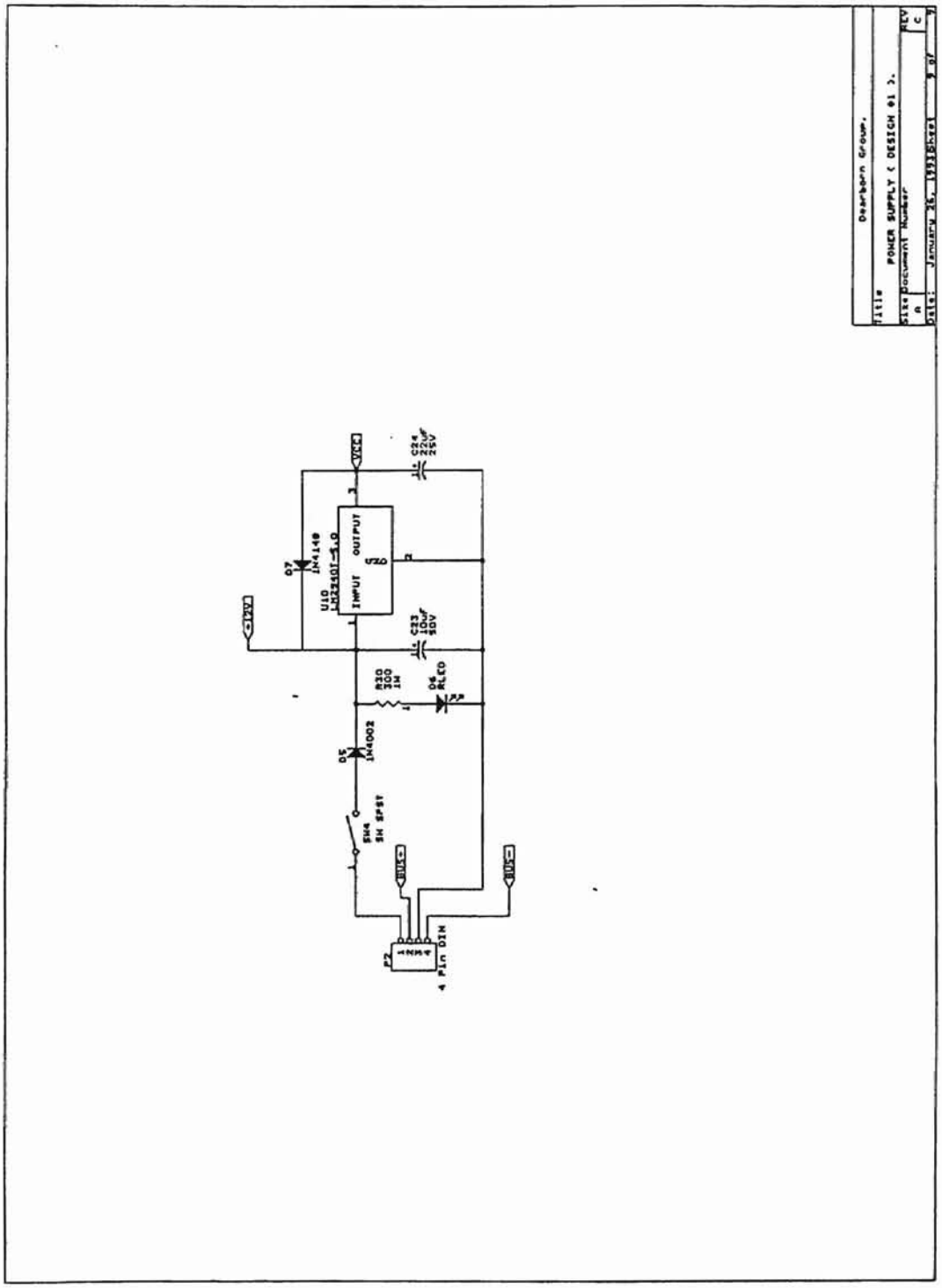

Dearborn Group
Dearborn Network Analysis Tool (PRU)
HEADER 36X2


TRIGGER
Input
RAM Protect
Switch (Ø = WE)
PC/STANDALONE
SELECT SW
(Ø = Standalone)
XMIT
LED
GREEN
RCV
LED
RED
BNC CONNECTOR
HEADER 20X2
HEADER 20X2
SW SPST
SW SPST
Dearborn Group
Dearborn Network Analysis Tool (I/O)


Dearborn Group,
Title
CCD INTERFACE TO 'HC11,
Size
A
Document Number
REV
Date: January 11, 1993
Sheet 4 of
JP5 JUMPER
JP6 JUMPER
JP7 JUMPER
R11 13K
R13 120
R15 13K
R12 10
R14 10
C10 3300pF 50V
C11 3300pF 50V
C12 .01µF RADIAL 50V
D3 1N5245
D4 1N5245
U7
CLK
VCC
A
B
MODE
XMIT BUS +
REC BUS -
SCK
CS*
IDLE*
CONTROL
GND
CCD
R9 10K
R33 10K
U11
NC
VCC
GND
OUT
4MHZ OSCILLATOR
BUS+
BUS-
VCC
GND
MISO
MOSI
SCK
IDLE*
CONT


Dearborn Group
Title
Chrysler Network Analysis Tool (RS-232)
Size
A
Document Number
REV
D
Date: February 16, 1993
Sheet 2 of
P1
CONNECTOR_DB9F
Reset
Transmit Select
J11
JUMPER
J10
JUMPER
Receive Select
J9
JUMPER
J8
JUMPER
C18
10uF
16V
VCC
C17
10uF
16V
C19
10uF
16V
C16
10uF
16V
U8
MAX232
T1OUT
R1IN
T2OUT
R2IN
V+
V-
T1IN
R1OUT
T2IN
R2OUT
C1+
C1-
C2+
C2-
J12
J13
GND


Dearborn Group.
RESET CIRCUIT
Reset Switch
R22
10K
VCC
RSTIN
MAX707CPA
MRESET
RESET
/RESET
VCC
GROUND
N.C.
PFI
PFO
SW3
N.O.
Reset
GND


Dearborn Group,
Title
POWER SUPPLY C DESIGN #1 2.
Size
A
Document Number
REV
C
Date:
January 26, 1999
Sheet
of
VCC
C24
22uF
25V
D7
1N4148
U10
LM2940T-5.0
INPUT
OUTPUT
+12V
C23
10uF
50V
R30
200
1W
D6
RLED
D5
1N4002
SW4
SW SPST
BUS+
BUS-
P2
1
2
3
4
4 Pin DIN

2

VITA

Ronald Edward Evans

Candidate for the Degree of

Master of Science

Thesis: DEVELOPMENT OF A NEURAL NETWORK BASED WEED SPRAYING SYSTEM

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Tulsa, Oklahoma, On April 27, 1969, the son of Robert Evans and Sharon Barnett.

Education: Graduated from Charles Page High School, Sand Springs, Oklahoma in May 1987; received Bachelor of Science degree in Electrical Engineering from Oklahoma State University, Stillwater, Oklahoma in May 1992. Completed the requirements for the Master of Science degree with a major in Electrical Engineering at Oklahoma State University in December 1995.

Experience: Graduate Research Assistant, Department of Biosystems and Agricultural Engineering, Oklahoma State University, February 1993 to July 1994. Teaching Assistant, Department of Electrical and Computer Engineering, Oklahoma State University, January 1992 to December 1993.