The Pennsylvania State University
The Graduate School
Department of Electrical Engineering

A DIGITALLY CONTROLLED SAFETY RADAR SUBSYSTEM FOR ATMOSPHERIC LIDAR SYSTEMS

A Thesis in
Electrical Engineering
by

Scott P. Boone

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 1995

_____
Scott P. Boone

We approve the thesis of Scott P. Boone.

Date of Signature

_____	_____
Charles R. Philbrick Professor of Electrical Engineering Thesis Advisor	
_____	_____
Stewart Kurtz Murata Professor of Materials Research, and Professor of Electrical Engineering	
_____	_____
William Moyer Research Associate	
_____	_____
Larry C. Burton Professor of Electrical Engineering Head of the Department of Electrical Engineering	

Abstract

The operation of high power lasers in the atmosphere requires the use of safety procedures to ensure the well being of ground personnel and the safety of passengers and crew aboard aircraft operating in the local area. For an atmospheric LIDAR to be used as an autonomous instrument, safety procedures in conjunction with a safety subsystem are required. This thesis describes the design and analysis of a Safety Radar subsystem using a commercial marine radar as its primary component. The subsystem includes custom hardware to monitor the radar and disable the laser when the situation is warranted. The Safety Radar subsystem is shown to detect both private and commercial air traffic at required ranges and at speeds which allow it to inhibit laser operation well before any intercept of the laser beam can occur. Self-monitoring functions for both the radar unit and the custom hardware are demonstrated to disable the laser in the event of any detected hardware failure. A summary of key features and cautions is given along with operational considerations.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express appreciation to my thesis advisor, Dr. C. R. Philbrick, for his encouragement and advice during the past year. I would like to thank Dr. Stewart Kurtz and Bill Moyer for serving on my thesis committee.

Many others have helped me through this long process, some offering technical assistance and others, wise counsel. Many of you just sat quietly while I "discussed" the technical issues written on these pages. This type of support is invaluable. To Mark Bregar, Dr. William Blum, Wayne Solbrig, Jack Woika, Jim Ross, Jeff Weber, and Jonathan Eckert, I offer a heartfelt thank you. The technical assistance of Tom Petach, David Blood, Dr. Daniel Lysak, Glenn Pancoast, Jeff A. Hamilton and Scott Kriner was also greatly appreciated.

A word of appreciation to my wonderful family, my wife, Lesa, my son, Tyler and baby-to-be, for their ongoing love and support and to my parents for their love and for making my education possible.

Chapter 1

Introduction

This thesis presents a detailed study of the development of the Safety Radar subsystem for use as a safety mechanism for atmospheric LIDAR (Light Detection and Ranging) systems. It shows the subsystem in a top-down fashion, starting with a subsystem definition followed by a parameter definition and analysis. It continues with a description of the functional design of basic hardware components, a discussion of subsystem testing, and concludes with a subsystem specification.

LIDAR systems direct a high power laser beam into the sky to measure profiles of atmospheric properties. The laser pulses, on the order of tens of nanoseconds duration, must be of high enough energy to produce measurable back-scattering from molecules and particles. Typical LIDAR applications use laser beams with 0.5 to 2 joules per pulse. These several hundred megawatt pulses are a hazard to the human eye for intraocular vision of the beam. Light passing through the lens of the eye is focused directly onto the highly sensitive retina. Only a small amount of optical energy directed at the eye will cause permanent retinal damage [1]. Experimental data shows that for a pulsed laser transmitting at 500 to 700 nm, retinal damage can occur at levels as low as 70×10^{-6} J/cm^2 [2]. Because of the optical hazard inherent in the high intensity laser pulse, strict safety procedures must be observed by operators of atmospheric LIDAR systems to avoid any direct intraocular vision of the beam.

With no precautions in place, aircraft could be illuminated by the laser causing unpredictable specular reflections from portions of the craft's surface. The reflection hazard,

along with the remote possibility of an aircraft passenger or pilot looking directly into the laser beam must be avoided. Safety procedures which inhibit the operation of the LIDAR system during periods when aircraft are near the laser provide one mechanism to ensure eye safety for aircraft crew and passengers, and ground based personnel.

The use of a human observer with a laser override switch is one solution to the safety problem. For short term operation of the laser, stationing one or two persons to observe aircraft activity provides the required safe operating environment, particularly in areas with modest air traffic. Observers are tasked with monitoring local air traffic and activating the laser override when an aircraft's flight path brings it into a predetermined area. The present operation of lasers in the vicinity of State College, Pennsylvania relies upon the use of safety observers. This operating procedure has been approved by the Federal Aviation Administration (FAA) for a site at the west end of the Pennsylvania State University campus. Similar operating procedures have been approved and used in many locations where laser beams are propagated in the atmosphere.

An automatic LIDAR system, one which would operate with minimal human interaction, has many potential applications. Such a system could monitor weather conditions in remote areas and transmit data to centralized sites. In many cases, this system could eventually replace the need for the launch of meteorological rockets and balloons [3]. In order to move the design of atmospheric LIDAR systems toward a goal of becoming an autonomous instrument, a reliable automatic safety subsystem must be developed. A radar based subsystem provides the search and detection capability required for this application.

Radar systems are used in many safety applications including air traffic control in busy flight corridors and ground-based collision avoidance systems. The role of a radar subsystem for an atmospheric LIDAR application is also one of safety. Instead of warning aircraft or ground based personnel of a potential hazard, the main requirement of this subsystem is to disable the operation of the laser beam well in advance of any aircraft intercept. Radar systems have been used in previous LIDAR instruments for the same purposes as those described here [4].

Chapter 2 defines the Safety Radar subsystem at a top level and shows the necessary parameters to describe it. Subsystem components are defined along with a discussion of how they relate to defined parameters. Tradeoffs for the high level components are discussed. Chapter 3 summarizes the top level requirements and describes how these requirements are satisfied by the subsystem. This section contains a detailed analysis of the subsystem's performance. Chapter 4 contains a detailed analysis of the custom hardware developed for the subsystem by the author and shows how it satisfies the requirements allocated to it. A discussion of the first generation of the hardware is included to further show the subsystem's present functionality. Chapter 5 includes discussions of proof-of-design and proof-of-performance tests that were performed on the subsystem and its components along with pertinent data taken during this testing. Chapter 6 concludes the thesis with a discussion of the subsystem's key features and a caution concerning the potential implementations of the subsystem. Operational considerations are discussed to summarize the uses of the

subsystem. An appendix is included containing hardware documentation, programming information for the subsystem, and MATLAB code used for analysis and calculations.

Chapter 2

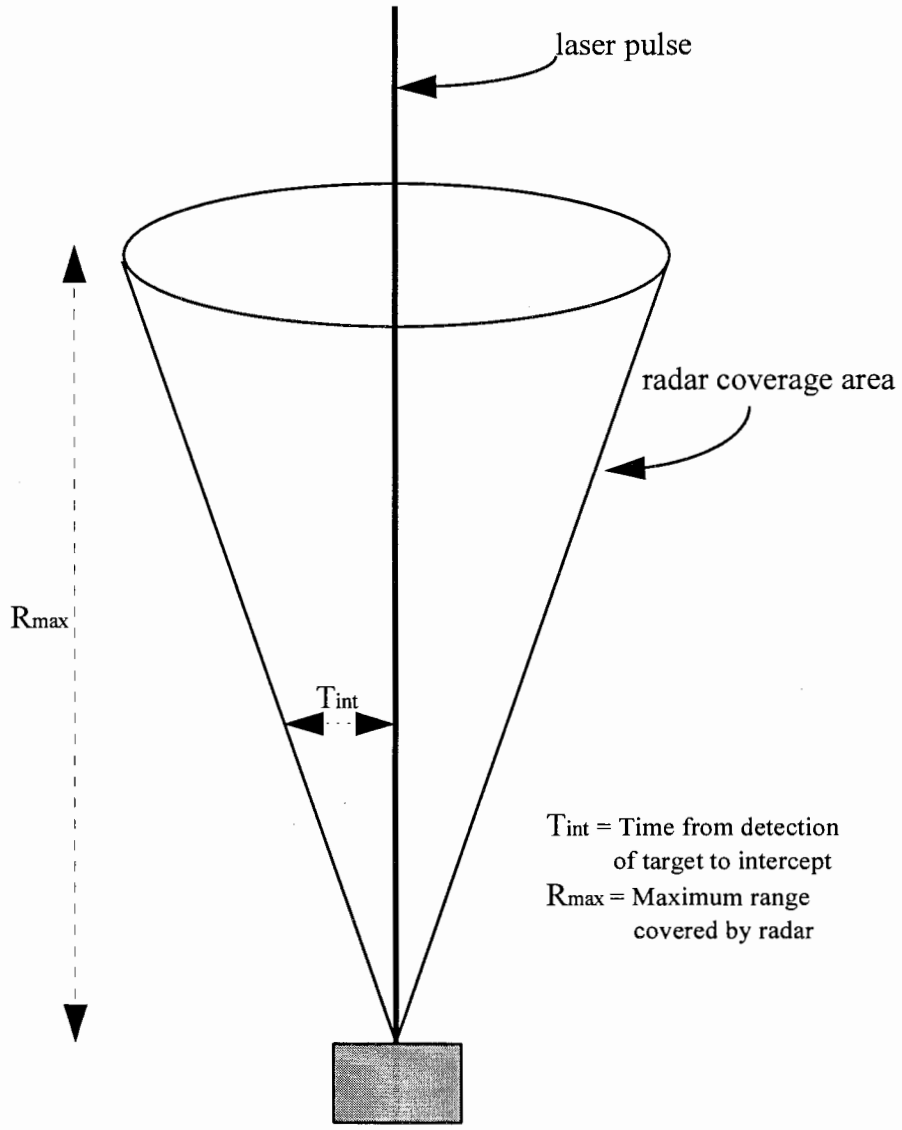
Subsystem Description

A radar subsystem to guard against potential intercepts of a laser beam must fulfill two basic requirements. Because lasers used in the current LIDAR systems have an eye safe distance of 180 km for intraocular vision [3], the radar subsystem must have sufficient range to detect aircraft up to their maximum operating altitudes or maximum range from the radar. It must also determine a potential intercept of the laser beam with sufficient speed and reliability to guarantee the laser is disabled and remains in that state until the hazard no longer exists. These requirements are discussed in general in this chapter to show the development of the overall subsystem.

Figure 2.1 illustrates the primary parameters that define the Safety Radar subsystem ; R_{max} and T_{int}. R_{max} is defined as the maximum range at which an aircraft is detectable by the subsystem. T_{int} is defined as the time from the first detection of an aircraft to the time it would intercept the laser. The subsystem must respond to a detection before this maximal parameter is reached. The Safety Radar subsystem's performance will be shown to meet necessary values for these parameters in the next chapter.

Commercially available radar systems offer the capability necessary to detect commercial and private aircraft at all possible altitudes and velocities. Several types of commercially available systems were evaluated including aircraft collision avoidance systems and marine radar systems. The latter provides the most cost effective solution while still maintaining needed performance. Because marine radar systems are in wide general use, they are easily maintained and repaired with readily available parts. The use of a marine

radar also produces some level of insensitivity to environmental changes. The radar set chosen for this application is a Raytheon R70 Series Marine Radar system.



Atmospheric LIDAR with Safety Radar Subsystem

Figure 2.1 Safety Radar subsystem parameter definition.

2.1 Commercially Available Hardware

The Raytheon Model R72 Raster Scan radar system is one example of a commercially available radar unit. It is a popular commercial and civilian marine radar. Its specifications are listed in Table 2.1 [5]. While the R72 provides adequate radar performance for this application, and is the major element in the subsystem, it alone does not provide the total solution. The sweep antenna common to this and most other marine radar systems does not lend itself to use in monitoring the area necessary to protect the airspace surrounding a laser beam. A parabolic antenna yields the desired conical area of coverage for the radar subsystem. Specifications for one such antenna are listed in Table 2.2 [6]. This antenna, model number AS15-90, is available from Seavey Engineering Associates. Its gain characteristic versus angle is shown in Figure 2.2.

Table 2.1 Raytheon R72 radar specifications [5].

Specification	Value
Transmitting Frequency	9410 +/- 30 MHz
Transmitting Power (peak)	10,000 W
Pulse Length @ Pulse Repetition Frequency (PRF)	0.08 μs @ 2000 Hz 0.25 μs @ 1500 Hz 0.7 μs @ 750 Hz 0.7 μs @ 500 Hz
Radar Receiver: Intermediate Frequency (IF)	60 MHz
Bandwidth	15 MHz @ 0.08 μs PRF 3 MHz @ all other PRF rates
Noise Figure	< 6 dB maximum

Table 2.2 Parabolic antenna specifications [6].

Specification	Value
Beam Width (half-power)	5.6 degrees
Efficiency@9400 MHz	60 percent
Gain (G)	30 dB

The Raytheon radar system with the modification of a parabolic antenna does not provide a complete solution for the design of the Safety Radar subsystem. The radar system is designed primarily to identify potential hazards for sea going vehicles. It includes a man-machine interface - a keypad for parameter entry and a video monitor to display potential targets in the radar's range. A direct link between the subsystem and the LIDAR system's laser firing circuit is necessary to control automatic shutdown of the laser anytime a potential intercept of the beam by an aircraft is detected. Custom designed hardware is used to interface the radar to the laser.

2.2 Custom Hardware - Laser Safety Radar Controller

Figure 2.3 illustrates a top level block diagram for the entire safety radar subsystem. The Laser Safety Radar Controller (LSRC) shown in the block diagram is the custom interface between the Raytheon R72 radar and the fire control for the LIDAR's laser. Chapter 4 is devoted to an explanation of the LSRC's background and detailed design. Indicators for the transmitted radar pulses and target returns are passed from the radar to the LSRC where they are examined and decisions are made about potential targets.

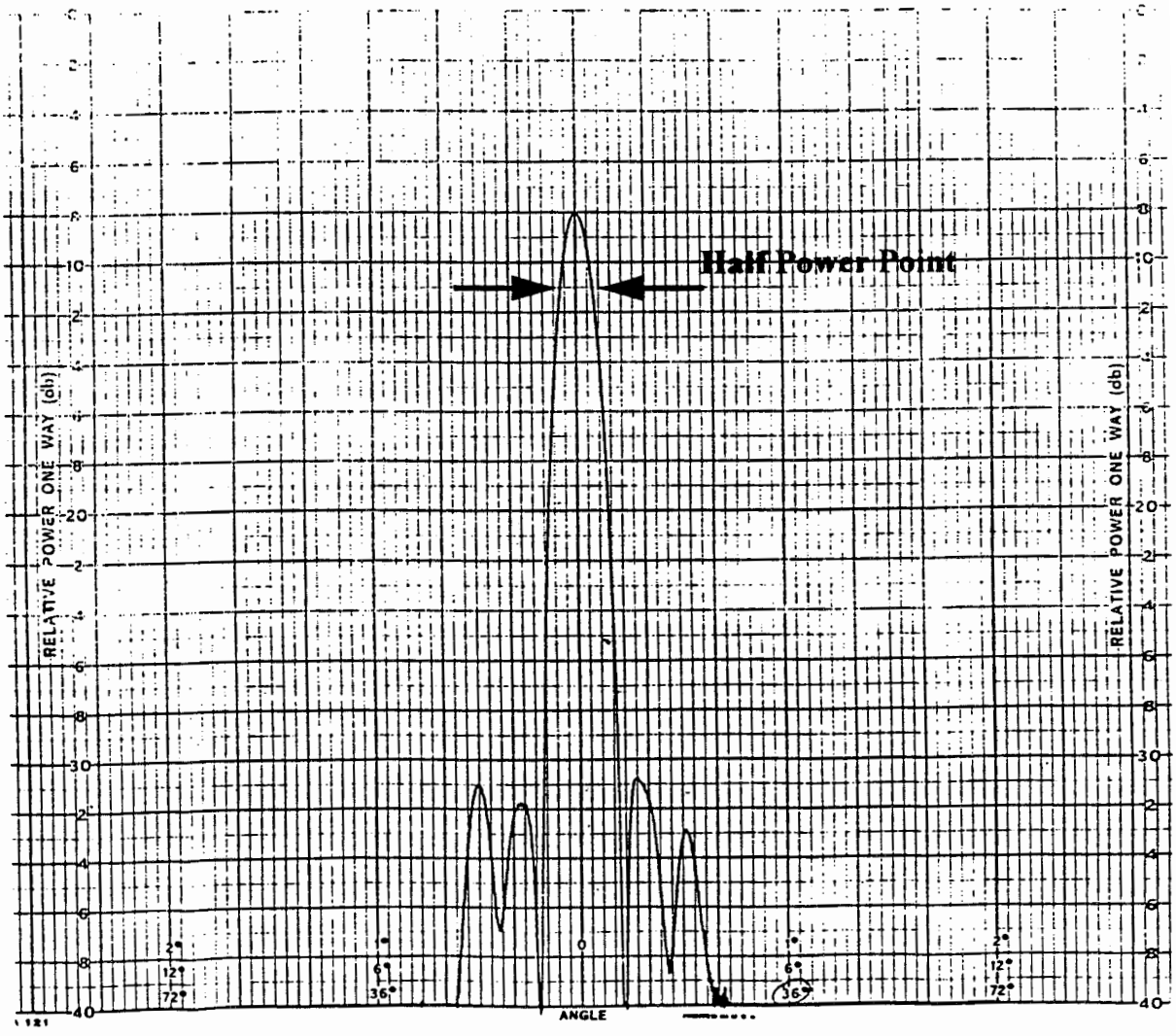


Figure 2.2 Seavey Engineering antenna, gain (dB) versus angle (degrees) [6].

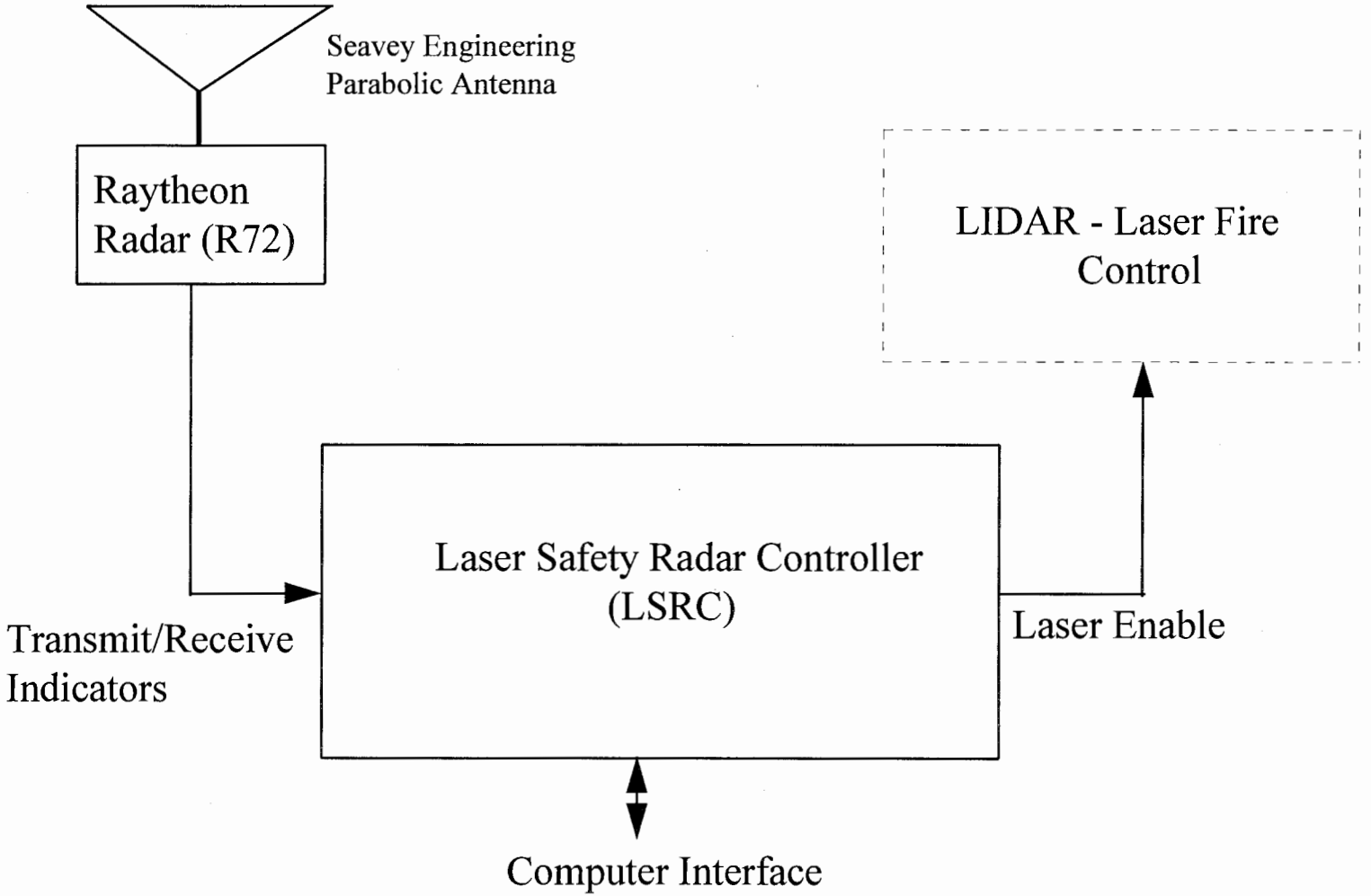


Figure 2.3 Safety Radar subsystem block diagram.

The LSRC supplies an enable signal to the LIDAR's laser control circuit. This enable must be present for the laser to be fired. Once identification of a potential target has been made or in the event of a hardware failure, the laser is disabled. A computer interface to the LSRC allows for operation of the entire subsystem using embedded control code. The LSRC performs self monitoring of several of its functions as well as functions of the R72. In the event of a detected failure of the LSRC or the R72 radar, the LIDAR's laser is disabled.

2.3 Safety Radar Subsystem

The Raytheon R72 radar, without the display interface or the delivered sweep antenna, are combined with the parabolic antenna and the LSRC defined above to make up the Safety Radar subsystem. The maximum detectable range parameter is directly related to the radar's transmitted power, receiver sensitivity and antenna gain specifications. It will be shown that the Safety Radar subsystem is capable of meeting or exceeding a value for R_{max} that safely covers commercial air traffic. The time to intercept parameter, T_{int} is related to both the radar's pulse repetition frequency (PRF) and the electronic response time of the LSRC. This parameter can also be viewed as a specification on the maximum velocity of an aircraft at a given altitude. The Safety Radar subsystem will be shown to have a probability of detection of commercial aircraft targets that approaches one. The analysis of these parameters is shown in Chapter 3.

Chapter 3

Subsystem Analysis

3.1 Aircraft Parameters

In order to analyze the performance of the Safety Radar subsystem, it is first necessary to discuss the parameters that describe the aircraft that it will detect. Table 3.1 lists the maximum speed and maximum certified service ceiling for three classes of aircraft [7]. The classes were chosen to provide a set of speeds and altitudes which are representative of the extremes encountered by the radar.

Table 3.1 Representative aircraft speeds and altitudes [7].

Aircraft Class	Maximum Level Speed km/hr (miles/hour)	Service Ceiling km (kft)
Small Private (Piper Cub)	210 (130)	5.8 (19)
Midsize Commercial (Boeing 737-200)	856 (530)	10.7 (35)
Large Commercial (Boeing 747-400)	981 (608)	13.7 (45)

An estimate of the radar cross section for each representative aircraft is also needed to prove subsystem performance. The radar cross section of a given radar target can vary widely based on the angle of incidence and transmitting frequency of the radar. The aspects which provides the minimum possible radar cross section to a radar are a nose view and a tail view. A top, bottom or broadside view represent the maximum radar cross section for a radar target. The Safety Radar subsystem is intended for use as a vertically pointing system with a half power beamwidth of 5.6 degrees. Commercial aircraft encountered by the radar therefore will be in relatively shallow dive or climb angles or in level flight relative to the

radar beam. Estimates of typical radar cross sections for the representative classes are given in Table 3.2 [8,9].

Table 3.2 Representative aircraft radar cross section estimates [8,9].

Aircraft Class	Radar Cross Section (square meters)
Small Private	1- 20
Midsize Commercial	20 - 40
Large Commercial	40 -100

These estimates shown in Table 3.2 represent minimum radar cross section values excluding nose and tail views. For the Safety Radar subsystem to encounter nose and tail aspects, aircraft would be required to dive or climb at extreme angles relative to the radar detection zone.

3.2 Maximum Range Analysis

The analysis of the maximum range for the Safety Radar subsystem is based on a minimum detectable signal level. A rule of thumb for RF receiver design is that a signal is regarded as detectable if it is at least 6 dB above the receiver noise floor. The thermal noise voltage in a receiver can be described statistically as a complex Gaussian process having a zero mean and a variance of half of the noise power [10]. A signal which is 6 dB higher than the noise or on the order of four times the power of the noise is considered to be detectable. To perform the maximum range analysis, the noise floor of the radar receiver is defined as

$$R_{noise} = kT + 10\log(BW) + NF, \qquad (3.1)$$

where:

kT = noise in a 50 ohm resistor at room temperature = -174 dBm/Hz,

BW = radar receiver bandwidth = 3 MHz,

NF = radar receiver noise figure = 6 dB [5].

For the Safety Radar subsystem's radar, R_{noise} = -103.23 dBm. Values for the minimum detectable signal above this noise floor, $S_{min,}$ are then considered in the calculation for the maximum range of the subsystem. This range is given by [8]

$$R_{max} = \sqrt[4]{\frac{PWR * G^2 * \lambda^2 * \sigma}{(4\pi)^3 * S_{min}}}, \qquad (3.2)$$

where :

PWR = radar transmitting power = 10000 W,

G = antenna gain = 30 dB,

λ= transmitter wavelength = 0.032 meters,

σ = radar cross section of the target in square meters.

Figure 3.1 and Table 3.3 show the results for the maximum range calculation of the Safety Radar subsystem versus the radar cross section of a target aircraft. The values for S_{min} considered are 6 dB, 9 dB, 12 dB and 15 dB. The MATLAB program used to generate the data is included in Appendix F. This calculation illustrates the maximum detection capability of the subsystem by using the full gain value of the antenna. An aircraft is shown to be detectable by the radar at altitudes much higher than those achievable by the representative aircraft for minimum detectable signals from 6 dB to 15 dB above the receiver noise floor.

Next it is necessary to show the detection ranges available to the radar at the half-power point of the antenna or 2.8 degrees off of the vertical axis. This will show the

detectable altitudes for aircraft at the outer edge of the detection zone. Note that while the minimum detectable signal values considered in this calculation remain the same, the gain value is 3 dB lower at the half power point of the antenna than at its main lobe. The results of this calculation are shown in Figure 3.2 and Table 3.4.

Table 3.3 Summary of maximum range calculation.

Radar Cross Section (square meters)	Max Range in km (kft):			
	S_{min} = 6 dB	S_{min} = 9 dB	S_{min} = 12 dB	S_{min} = 15 dB
1	12.8 (42)	10.8 (35)	9.1 (30)	7.6 (25)
5	19.2 (63)	16.1 (53)	13.6 (45)	11.4 (37)
10	22.8 (75)	19.2 (63)	16.1 (53)	13.6 (45)
20	27.1 (89)	22.8 (75)	19.2 (63)	16.1 (53)
40	32.2 (106)	27.1 (89)	22.8 (75)	19.2 (63)
100	40.6 (130)	34.1 (112)	28.7 (95)	24.2 (79)

Table 3.4 Summary of half power range calculation.

Radar Cross Section (square meters)	3 dB Range in km (kft):			
	S_{min} = 6 dB	S_{min} = 9 dB	S_{min} = 12 dB	S_{min} = 15 dB
1	9.1 (30)	7.6 (25)	6.4 (21)	5.4 (17)
5	13.6 (45)	11.4 (37)	9.6 (32)	8.1 (27)
10	16.1 (53)	13.6 (45)	11.4 (38)	9.6 (32)
20	19.2 (63)	16.1 (53)	13.6 (45)	11.4 (37)
40	22.8 (75)	19.2 (63)	16.1 (53)	13.6 (45)
100	28.7 (94)	24.2 (79)	20.3 (67)	17.1 (56)

Table 3.4 shows that for a minimum detectable signal that is 6 dB above the receiver noise floor, a representative aircraft is detectable at 2.8 degrees off of the vertical axis to more than

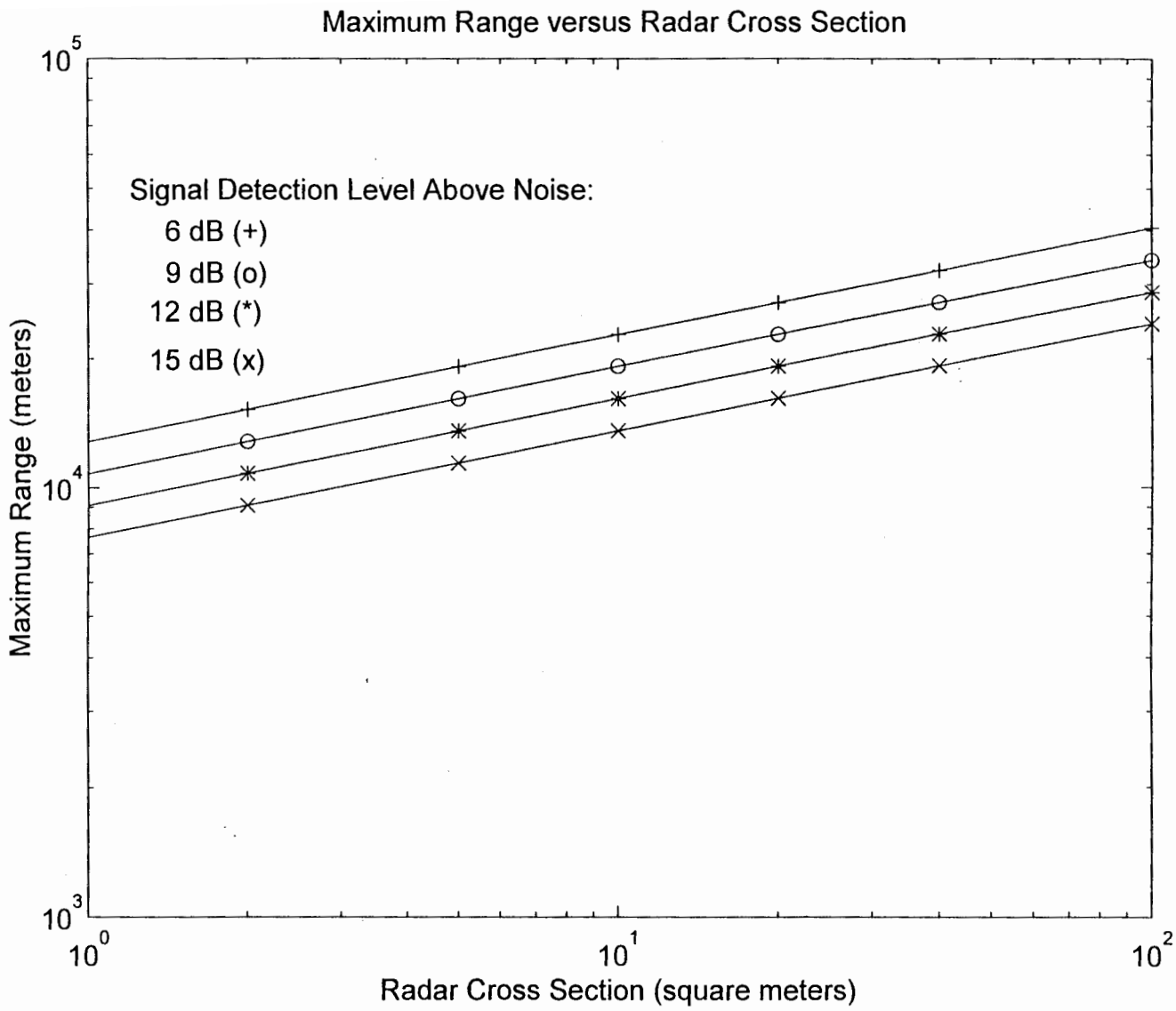


Figure 3.1 Maximum range versus radar cross section.

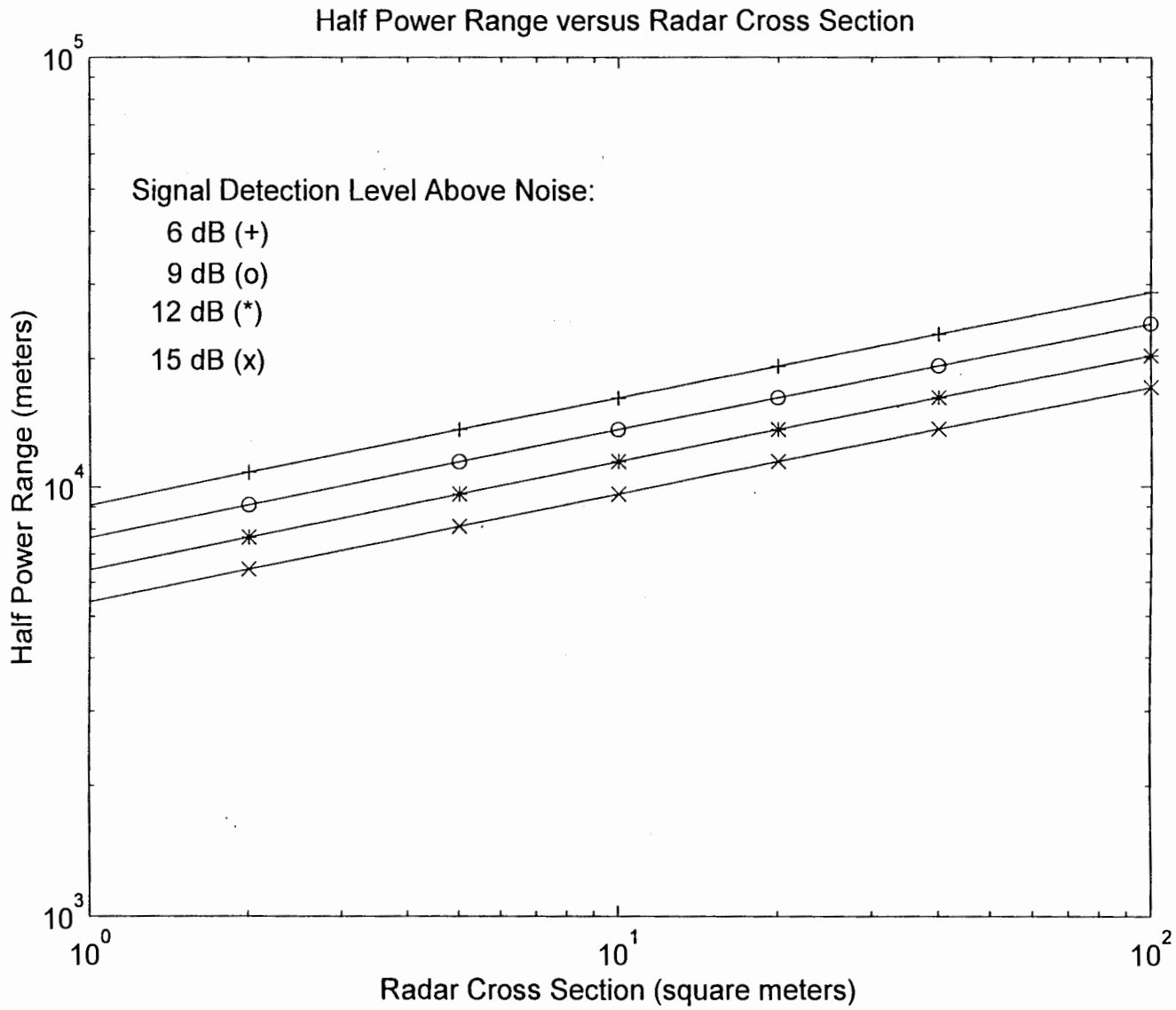


Figure 3.2 Half power range versus radar cross section.

1.5 times the required altitude. Minimum detectable signals up to 12 dB above the receiver noise floor surpass the required detection altitudes for all representative aircraft. Note that because the antenna is used to transmit and receive radar signals, a 3 dB reduction in antenna gain yields an overall reduction in the calculation of 6 dB.

3.3 Maximum Time to Intercept Analysis

Having shown that the SRS has sufficient power to detect the representative aircraft at their service ceilings, it now must be shown that this detection can take place in a sufficiently short amount of time to guarantee that the LIDAR is disabled in advance of any intercept of the laser beam. The basic problem is illustrated in Figure 3.3.

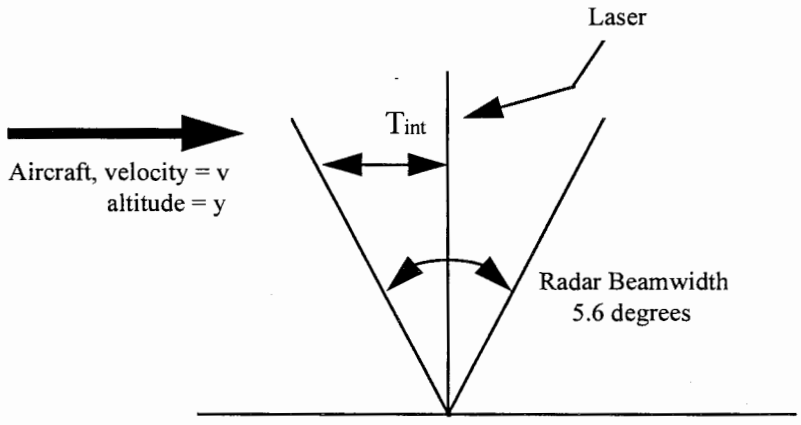


Figure 3.3 Calculation for time from detection to intercept, T_{int}.

The PRF or pulse repetition frequency of the radar is an important parameter for this problem. Of the available PRF's for the Raytheon radar, the 750 Hz setting delivers the

maximum average power. This is the highest frequency setting that delivers the longest available pulse (700 ns). Other settings using higher PRF rates and shorter pulses are available from the radar and are intended for shorter range operation. It should be noted that at the 750 Hz PRF setting, the radar cannot detect targets that are at altitudes under 450 meters due to the radar receiver's response to the radar's transmission of its pulse. This will be explained in more detail in Chapter 5. Targets down to 300 meters are detectable using the 1500 Hz PRF setting which transmits a 250 ns pulse.

It can be seen from Figure 3.3 that the minimum velocity of an aircraft to intercept the laser without being detected is given by the following formula:

$$v_{min} = y * \tan \theta / T_{int} \qquad (3.3)$$

The angle θ is half of the 3 dB beamwidth of the parabolic antenna or 2.8 degrees. For example, if T_{int} is taken as ten 750 Hz periods or 13.3 milliseconds, then the minimum velocity necessary to intercept the beam without the aircraft being detected is 7923 km/hr. Table 3.1 gives the maximum level speed of the fastest aircraft considered as 981 km/hr. At this velocity and an altitude of 300 meters, the value for T_{int} is on the order of 50 milliseconds. Considering this extreme case of an aircraft cruising in excess of 900 km/hr at a 300 meter altitude, the radar would has more than 35 pulses at the 750 Hz PRF with which to detect it. When considering altitudes above 300 meters and/or representative aircraft velocities less than 900 km/hr, an increasing number of radar pulses is available due to the increased value of T_{int}.

It is necessary to determine then, how many consecutive radar returns from an aircraft will be required to designate a shutdown condition for the laser. This involves a tradeoff between the probability of detection and the probability of false alarm. Every radar receiver has a finite amount of noise associated with it creating a statistical process for the detection of any one radar return. The probability of detection for an individual radar pulse is determined by examining several parameters. By determining the signal to noise ratio of the radar return pulse and choosing a probability of false detection that is acceptable to the application, a probability of detection for a single pulse can be determined [9,10,12].

For example, Figure 3.4 shows that for a 12 dB SNR signal and a probability of false detection of 10^{-6}, the probability of detection of a single pulse is 0.7. A cumulative probability of detection may be determined by examining more than one return from the radar target. To achieve the cumulative probability of detection for a target which requires that any one of N radar returns is detected, the following equation is used [11].

$$p_c = 1 - (1 - p_d)^N \tag{3.4}$$

where:

p_c = probability of cumulative detection,
p_d = probability of detection of one pulse,
N = number of returns.

It can be seen from Equation 3.4 that as more pulses are reflected from a target, the probability of detection of that target increases. Figure 3.5 shows an example calculation for Equation 3.4 for a cumulative detection probability of 0.99999. With N = 35 and a

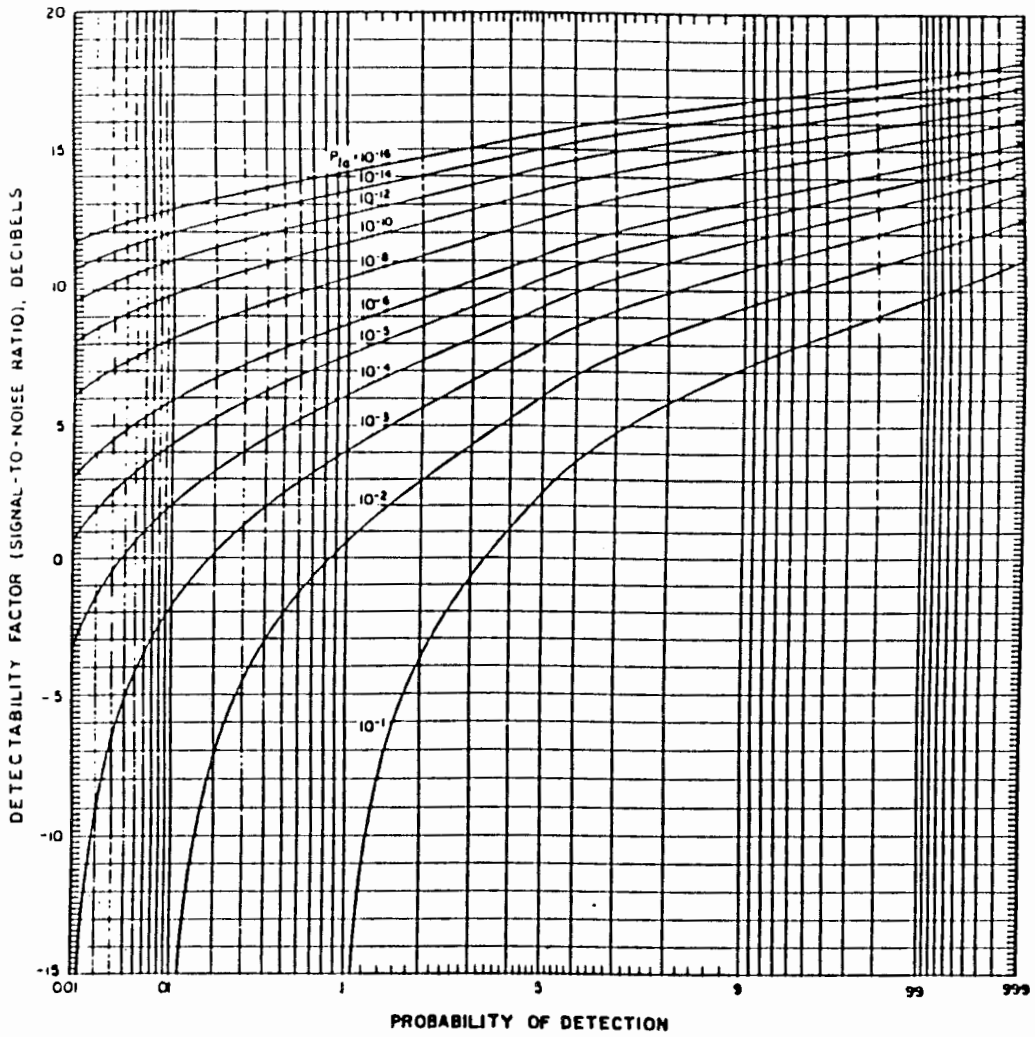


Figure 3.4 Required signal-to-noise ratio for single-pulse detection [9].

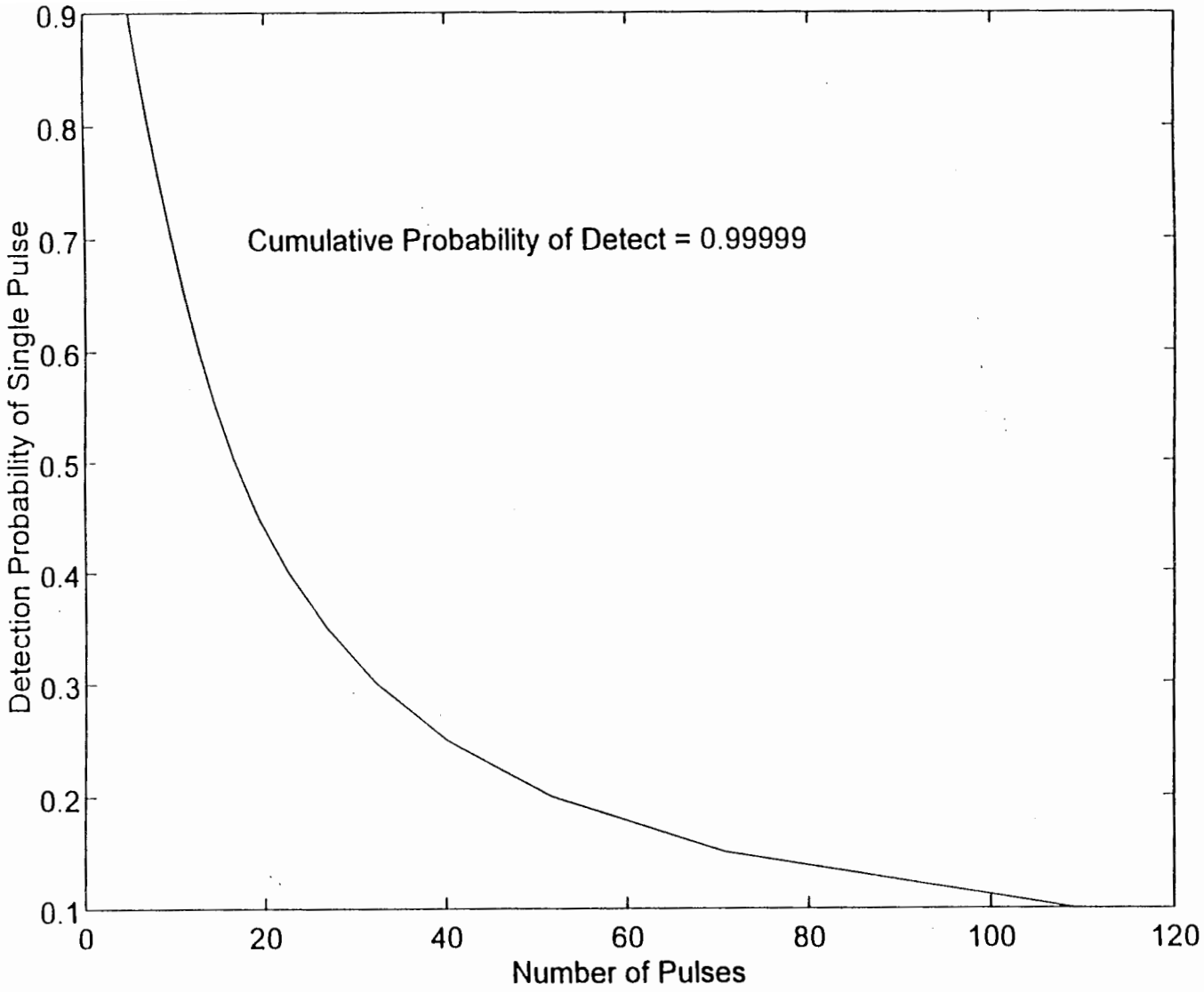


Figure 3.5 Probability of detection for a single radar pulse versus the number of pulses returned from the target for a cumulative probability of detect of 0.99999.

probability of detection of one pulse of 0.7, any of the representative aircraft will be detected with a probability approaching 1.0. If only 10 pulses are received with a single pulse detection probability of 0.7, the cumulative probability of detection is still 0.999994.

The probability of false detection for the previous example is prohibitively high. To reduce the probability of false detection it is necessary to examine more than one consecutive return pulse. This technique is referred to as pulse integration. Figure 3.6 shows how the probability of false alarm for a given probability of detection is improved by the integration of pulses received by a linear detector. The combined probability of detection of two or three consecutive radar return pulses lowers the probability of detection for the combined event. For example, with a probability of detection of 0.7 for a single pulse, the probability of detecting two consecutive pulses is 0.49 or the product of the probability of detection for each single pulse. Similarly, three consecutive pulses yields a probability of detection of 0.343. For the previous example which considered 10 radar pulses, the cumulative probability of detection for two consecutive pulses is 0.998. When considering a 12 dB SNR signal, a probability of false detection of less than 10^{-10} is then achieved. Similarly, considering 10 pulses for three consecutive pulses yields a cumulative probability of detection of 0.965 with a probability of false detection of less than 10^{-12}. When considering the large number of pulses available on a target as discussed earlier, the lowered detection probability for each event is of no consequence. The integration of two or three radar return pulses by the Safety Radar subsystem will still yield a detection probability approaching one because the number of pulses available for integration was shown to be at least 35.

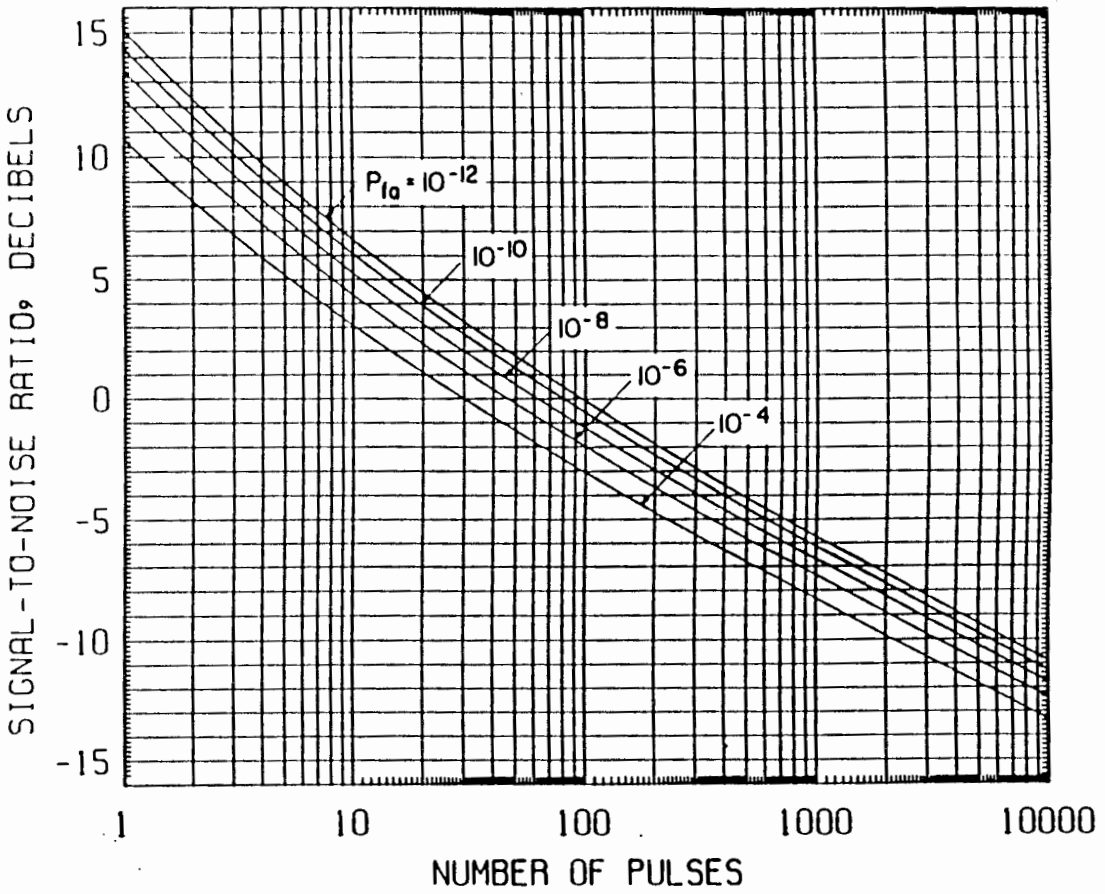


Figure 3.6 Required signal-to-noise ratio for detection with noncoherent integration of pulse; linear detector, nonfluctuating signal, p_d =0.75 [9].

3.4 Analysis Summary

The preceding analysis shows that the Safety Radar subsystem, with the Raytheon R72 as its primary component, meets or exceeds necessary parameters to protect aircraft from intercepting the laser in the atmospheric LIDAR system. Maximum range parameters were shown to be met for all representative aircraft. It was shown that the aircraft are detectable well within the necessary time prior to a potential intercept of the laser beam. The probability of detection of a target was shown to approach one for the representative aircraft.

Aircraft flying at altitudes above 300 meters and up to a Boeing 747-400's maximum level cruising speed, are detectable and the laser will be disabled well in advance of any laser beam intercept.

To demonstrate the Safety Radar subsystem as a reliable system, it is necessary to address other effects known to influence the detection of radar signals. Signals transmitted at microwave frequencies are susceptible to absorption by clouds and scattering by rain. At the subsystem's transmit frequency of 9.4 GHz, absorption by clouds accounts for only a loss of on the order of 0.01 dB/km. Scattering by rain at this frequency can account for a more significant loss in signal. Table 3.5 gives characteristic values of loss caused by scattering effects based on the rate of rainfall [11]. It should be noted that at rainfall rates over 25 mm/hr (1 in/hr), the range performance of the subsystem becomes significantly affected. The Doppler effect is of no consequence for this application due to the 3 MHz bandwidth of the receiver. A radar target would be required to have a velocity relative to the radar greater than 86078 km/hr to shift the received return pulse out of the band of the receiver.

Table 3.5 Scattering loss of 9.4 GHz signal transmitted in rain [11].

Rate of Rainfall, mm/hr (in/hr)	Scattering Loss, dB/km
1.25 (0.05)	0.02
5 (0.2)	0.09
25 (1)	0.9

The above analysis places the following requirement on the custom hardware as denoted in Figure 2.3 as the Laser Safety Radar Controller (LSRC). The LSRC must detect shutdown conditions by performing pulse integration of up to three pulses and disabling the laser within 1 millisecond when such a condition is found. This requirement is met by the LSRC as shown in Chapter 4. In addition, self monitoring features were added to the hardware to disable the laser in the event of a detected failure of a subsystem component.

Chapter 4

Custom Hardware Design Description

4.1 Original Design

In June of 1993, the original version of the Laser Safety Radar Controller (LSRC) was designed and constructed by W. Moyer of the Applied Research Laboratory at the Pennsylvania State University [14,15]. This design monitored two signals from the radar denoted as TRIG and VD. TRIG is the main radar timing signal. It generates its characteristic waveform as shown in Figure 4.1 for each radar pulse that is transmitted. The second signal monitored is VD, which is the detected radar signal utilized by the radar's display to indicate targets. Figure 4.1 shows the relationship for TRIG and the actual transmission of the radar. The parameters listed for the recurrence of TRIG and the pulse width of the radar transmission are obtained using the 750 Hz PRF setting of the R72 radar. The discussion that follows on the original LSRC design is described completely by Moyer [14,15].

The original radar interface circuit is described in a block diagram in Figure 4.2. It was designed as a circuit card mounted in close proximity to the R72 radar unit. The circuit detects transitions of the TRIG signal and the VD signal. The transitions of the TRIG signal are regular as depicted in Figure 4.1. The transitions of the VD signal depend on what targets the radar signal encounters. The signal is normally at zero volts and swings negative to a maximum value of -4 volts when a target is detected. In addition, at approximately 500 ns after the radar begins the transmission of its pulse, a negative pulse appears on the VD signal. This pulse is on the order of 4 μs in length. A plot showing the relationship of the pulse on

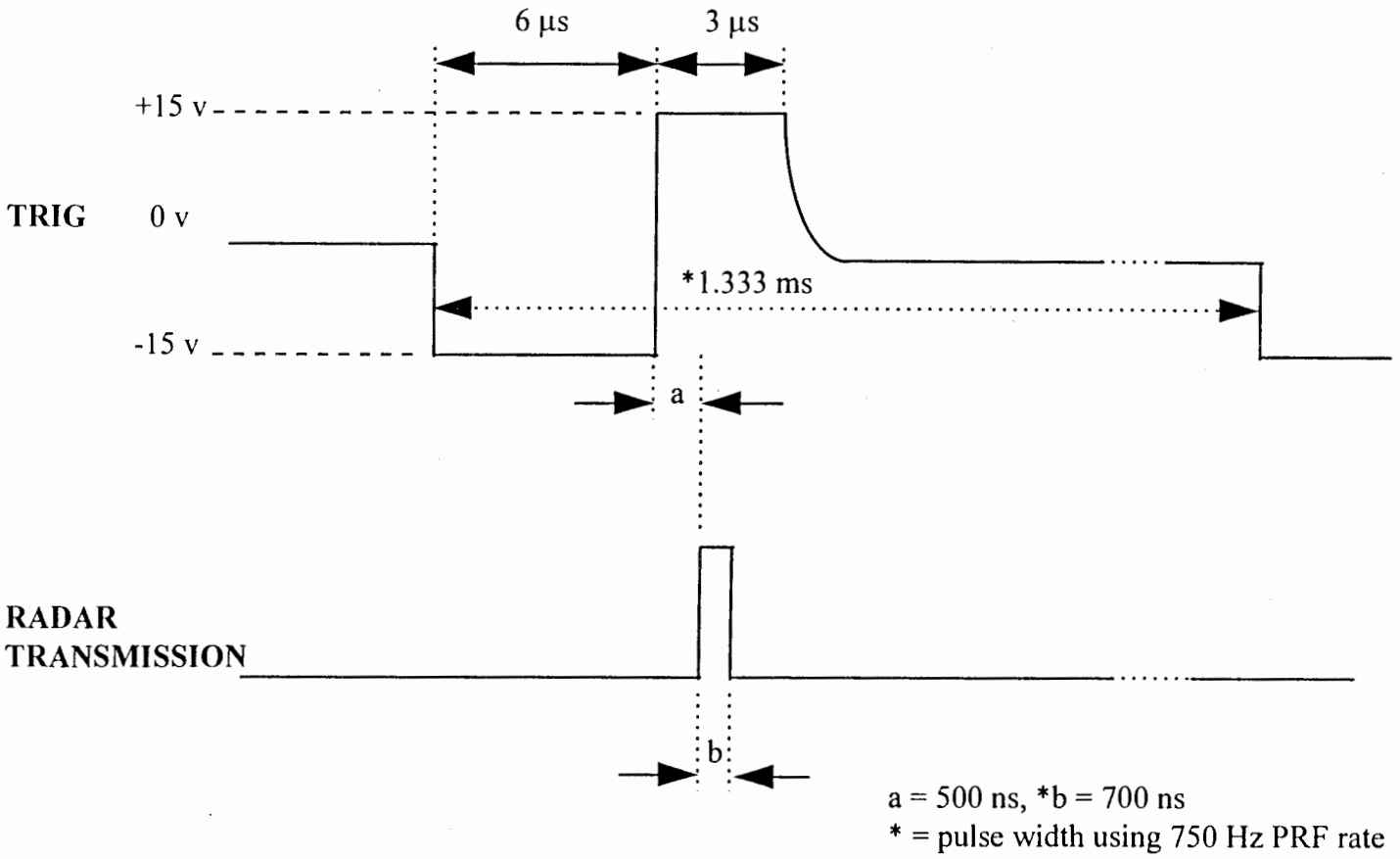


Figure 4.1 Relationship of radar TRIG signal to actual radar transmission.

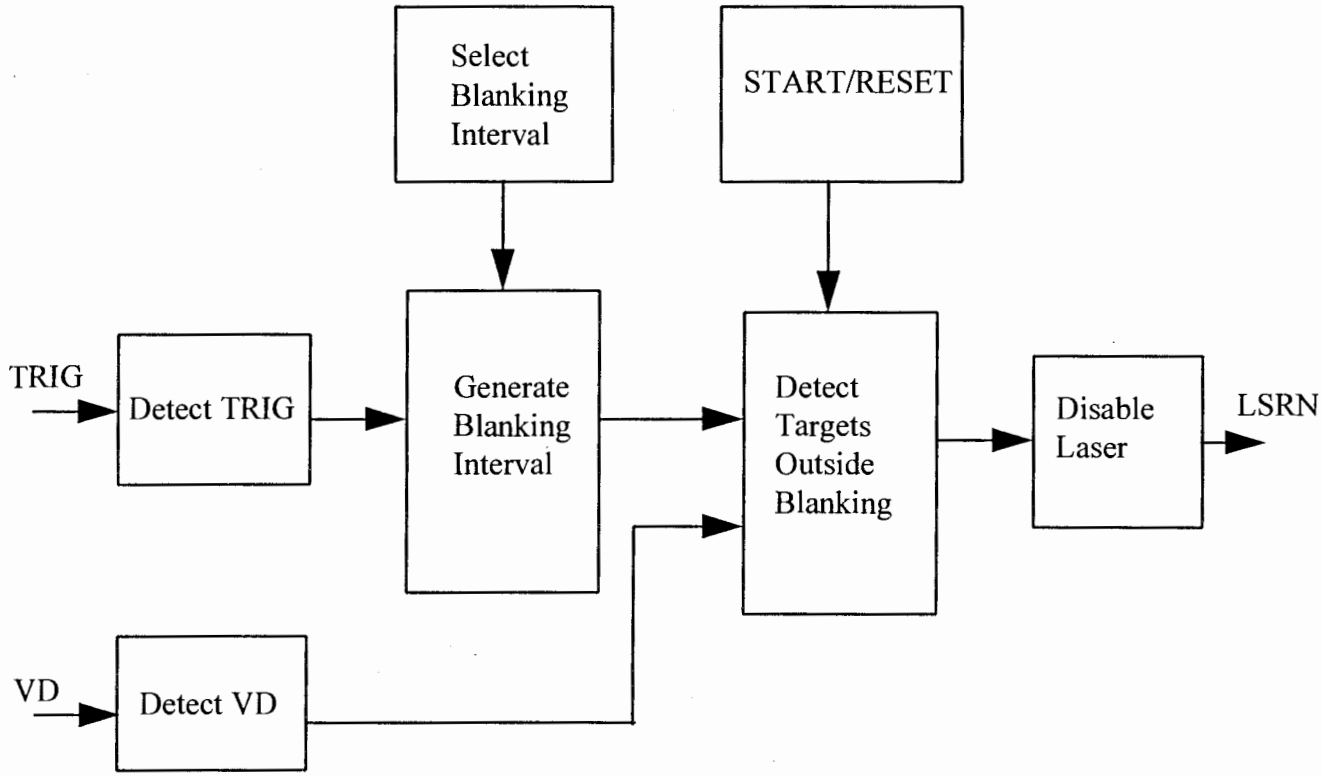


Figure 4.2 Block diagram of original laser safety controller circuit.

VD to the TRIG signal is shown in Figure 4.3. This pulse is caused by the transmission of the radar signal and the changeover of the radar's front end hardware from a transmit mode to a receive mode. The switching is necessary to protect the sensitive radar receiver from the high power transmit pulse. A blanking function is implemented in the hardware to allow for the this first transition of the VD signal to be ignored. The amount of blanking selected for the circuit is set up through manual switch settings located on the circuit card.

The circuit monitors the VD signal beyond the selected blanking interval and stores transitions that occur. If a transition that exceeds a fixed negative threshold is found on the VD signal following two consecutive TRIG signals, the laser is disabled. The control of the laser is accomplished through the LSRN signal. This optically coupled, TTL level signal must be in a logic one state (greater than 2.8 volts) for the laser to be enabled. After power up of the circuit or following a shutdown condition, a START/RESET switch must be activated to continue operation of the LSRC and enable the laser if no further shutdown condition is detected.

As noted by Moyer [14], the original circuit is relatively simplistic. This aside, it proved the concepts of interfacing to the radar signals and issuing a disable signal when a target was detected. During field tests, it was found that the target detection method used by the circuit is susceptible to interference from other radar units operating nearby. Although some pulse averaging was accomplished, no target range information is considered in the decision to disable the laser. Two noise pulses on the VD signal that exceed the fixed

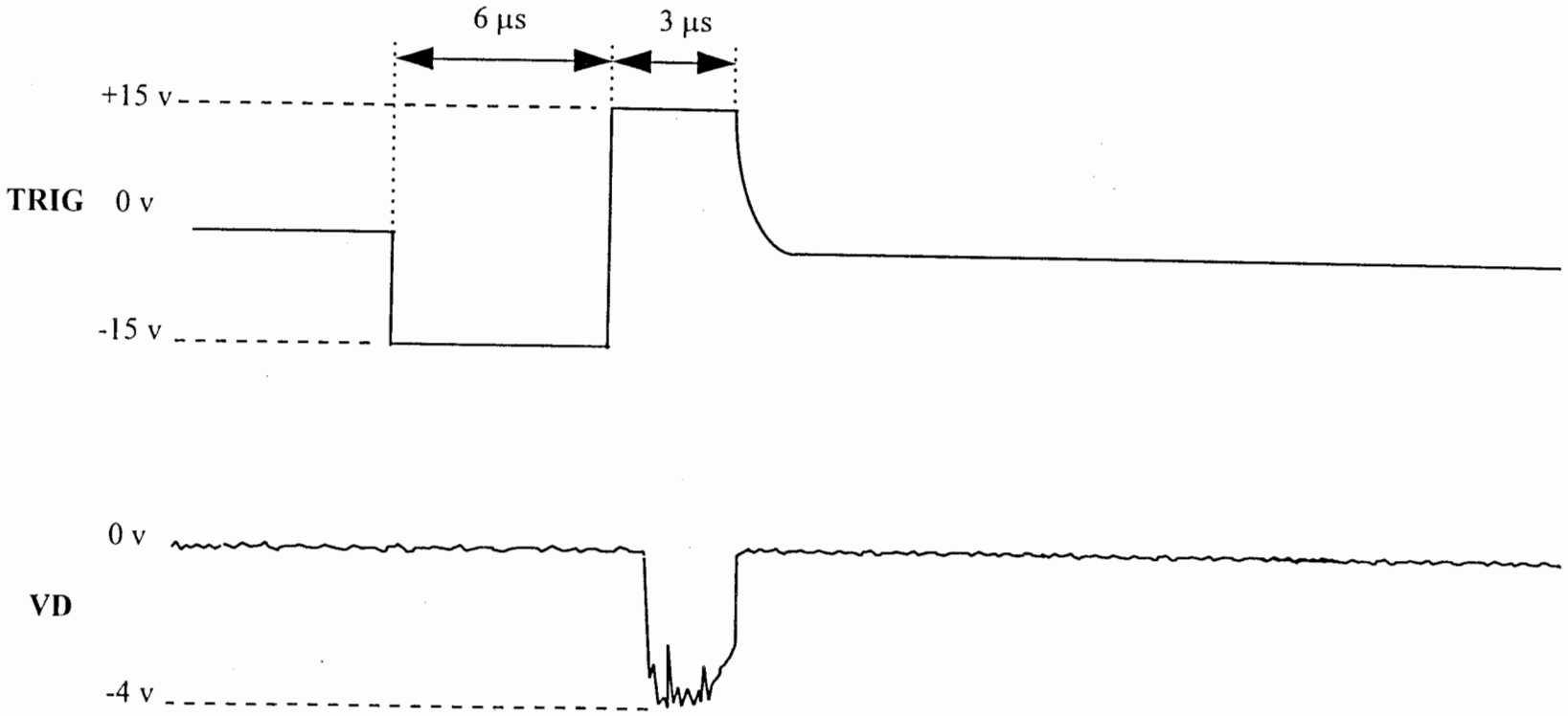


Figure 4.3 Relationship of radar VD signal to radar TRIG signal.

negative threshold at different times relative to the TRIG signal will invoke a shutdown of the laser. The circuit also lacked a computer interface for flexible control of the circuit and monitoring of status. It was determined that a revised LSRC was needed to overcome the limitations of the original.

4.2 Design Overview

The basic design of the original LSRC is incorporated in the new hardware. Figure 4.4 shows the block diagram of the processing of the detected radar target signals for the current LSRC. The detection methods for the TRIG and VD signals are a direct copy from the original design. These circuits implement high speed voltage comparators and needed no improvements or modifications from the original design. The features added to the original LSRC include a computer interface for setup, control, and status information, built-in testing functions (BIT), dynamic variation of the threshold used to detect the VD signal, implementation of a range bin algorithm for target detection and an on card display for visual reference of the current status.

A separate feature of the LSRC was added to the design to make the Safety Radar subsystem less susceptible to other radar units operating at the similar frequencies and PRF rates in the same locale. This feature is referred to as a pulse staggering or PRF jittering [11]. The block diagram for this function is shown in Figure 4.5.

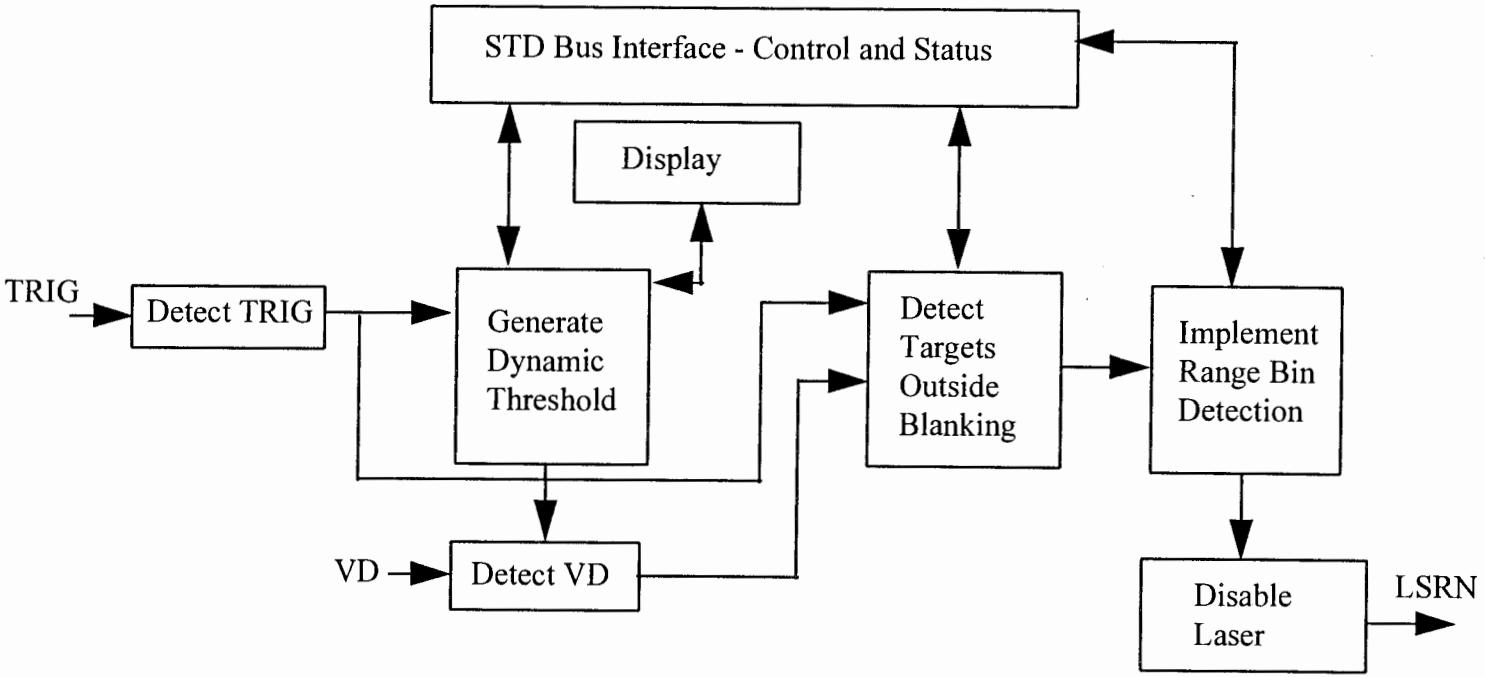


Figure 4.4 LSRC block diagram - main processing.

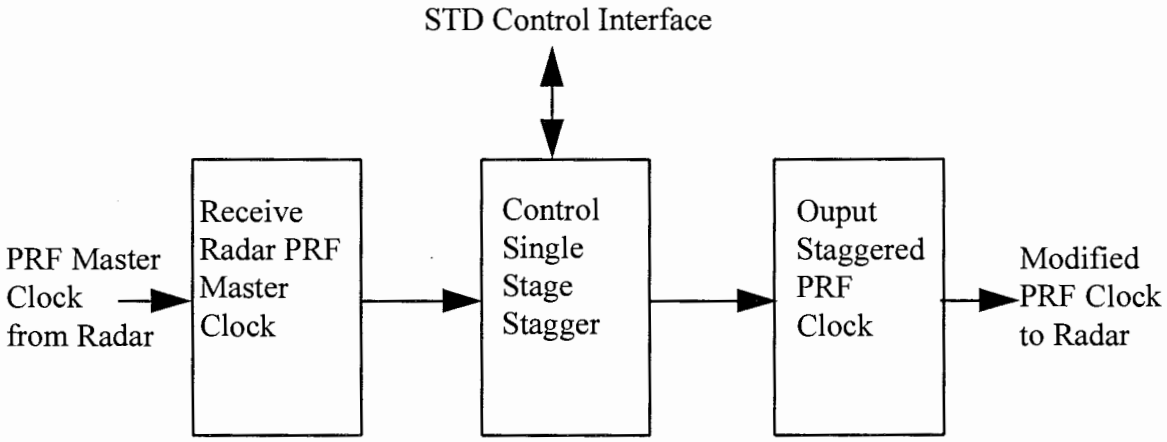


Figure 4.5 LSRC stagger controller block diagram.

4.3 Functional Description

The LSRC incorporates synchronous circuit design. The main clock is a 16 MHz, TTL level oscillator which drives a clock divider circuit implemented in a programmable logic device (PLD). The following sections describe the details of the major functional blocks of the LSRC. The schematics for the circuits described are located in Appendix A. Also included in the Appendices are a parts list for the circuit card assembly, a hardware/software interconnect diagram, the programmable logic code, written in ABEL, for the PLDs used and source code written in the C programming language used to generate detection threshold data for a programmable read-only memory (PROM).

4.3.1 Radar Pulse Stagger Controller

Modifying the characteristic PRF of the radar in the Safety Radar subsystem causes the timing of the transmitted and received pulses to differ from other radar units operating

with similar parameters. This makes the pulses transmitted more easily distinguishable from those transmitted by other radar units. The pulse stagger concept is illustrated in Figure 4.6.

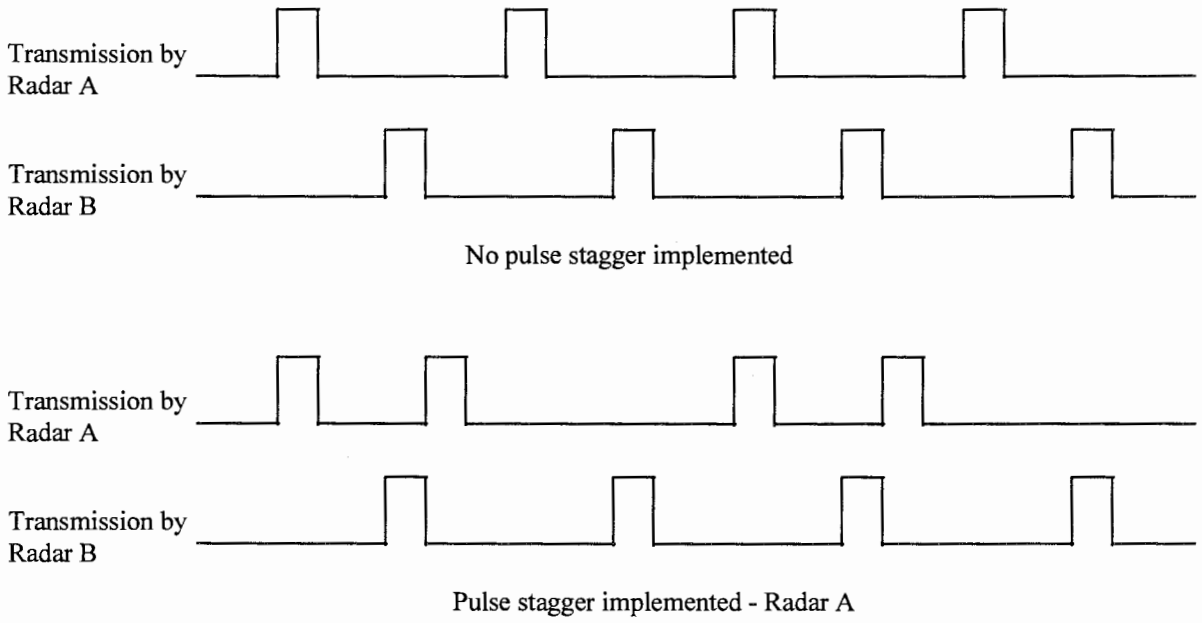


Figure 4.6 Pulse staggering of a radar transmission with alternate long and short intervals between pulses.

In the case shown with no pulse stagger implemented, Radar B appears as a target at a fixed range to Radar A. By delaying every other transmission of Radar A by a fixed amount, the false target shown to Radar A by Radar B changes apparent range enough that it will not be detected as a target by the range bin controller described in Section 4.3.3. By selecting a delay time for every other radar pulse of 2 μs with a 750 Hz PRF, the false target produced by Radar B appears to Radar A to shift by 300 m in 1.33 ms. A real target causing this effect would be traveling at 225000 m/s. With the stagger algorithm, use of a two pulse detection

method should eliminate false target acquisitions due to other radars with operating characteristics similar to the R72.

To implement the stagger controller in the subsystem it is necessary for the LSRC to alter the PRF rate generated by the R72 prior to its transmission. A careful analysis of the schematic for the display unit of the R72 reveals a jumper position that allows for it to be triggered by an external source. In an effort to keep the R72 signals isolated from the LSRC signals, opto-isolators are used to both receive the R72's master PRF clock and output the staggered PRF clock. The electrical interfaces used were suggested by the opto-isolator manufacturer for the device families present on the display unit of the R72.

The radar pulse stagger algorithm is implemented in a PLD. A delay of 0-6 μs in 2 μs increments is implemented. The delay selection is accomplished through 2 control bits from the computer interface. The input master PRF clock is a 50 percent duty cycle, 750 Hz clock. The PLD implementing the stagger algorithm detects this input with a 500 kHz clock by double buffering and edge detecting it to overcome any possible metastability problems associated with detecting an asynchronous signal. This process induces an additional pseudo-random jitter of the output pulse of up to 2 μs. The final output will delay the first of two pulses by 0-2 μs and the second of two pulses by 2-8 μs as dictated by the selection value. The stagger function may be disabled completely by disconnecting it from the R72.

4.3.2 Target Detection Threshold Controller

The original design of the LSRC used a fixed threshold voltage to detect signals on the detected radar target signal VD. To increase the sensitivity of the detection of long range targets, a dynamically varying threshold circuit was implemented.

Figure 4.7 shows a block diagram for the circuit. The VD signal from the radar is nominally zero volts and falls negative to -4 volts when a target is detected. VD is routed through a voltage divider circuit and then into a voltage comparator and is detected if it falls to a threshold value set at the comparator's input. The voltage divider biases the signal to 2.5 volts when VD is zero. The output of the voltage divider drops by half of the negative voltage excursion of VD. As an example, an excursion on the VD signal of -1 volt causes the output of the voltage divider to fall from 2.5 volts to 2 volts. Measurements of the VD signal have shown up to +/-500 mV of noise. To avoid false detection due to noise transients, the maximum threshold value for the comparator considered for detection is 2.1 volts. VD signals falling to within 5 mV of the threshold are detected and a corresponding TTL level pulse is output from the voltage comparator.

A control bit from the computer interface controls whether a fixed or a dynamic threshold is used for VD detection. If a fixed threshold is selected, the controller continuously outputs the first address of the PROM which contains the desired detection level. This level is currently set at 1.9 volts.

If a dynamic threshold value is selected, the controller, after receiving the detected TRIG signal, issues an address to a PROM containing threshold values every 2 μs. The

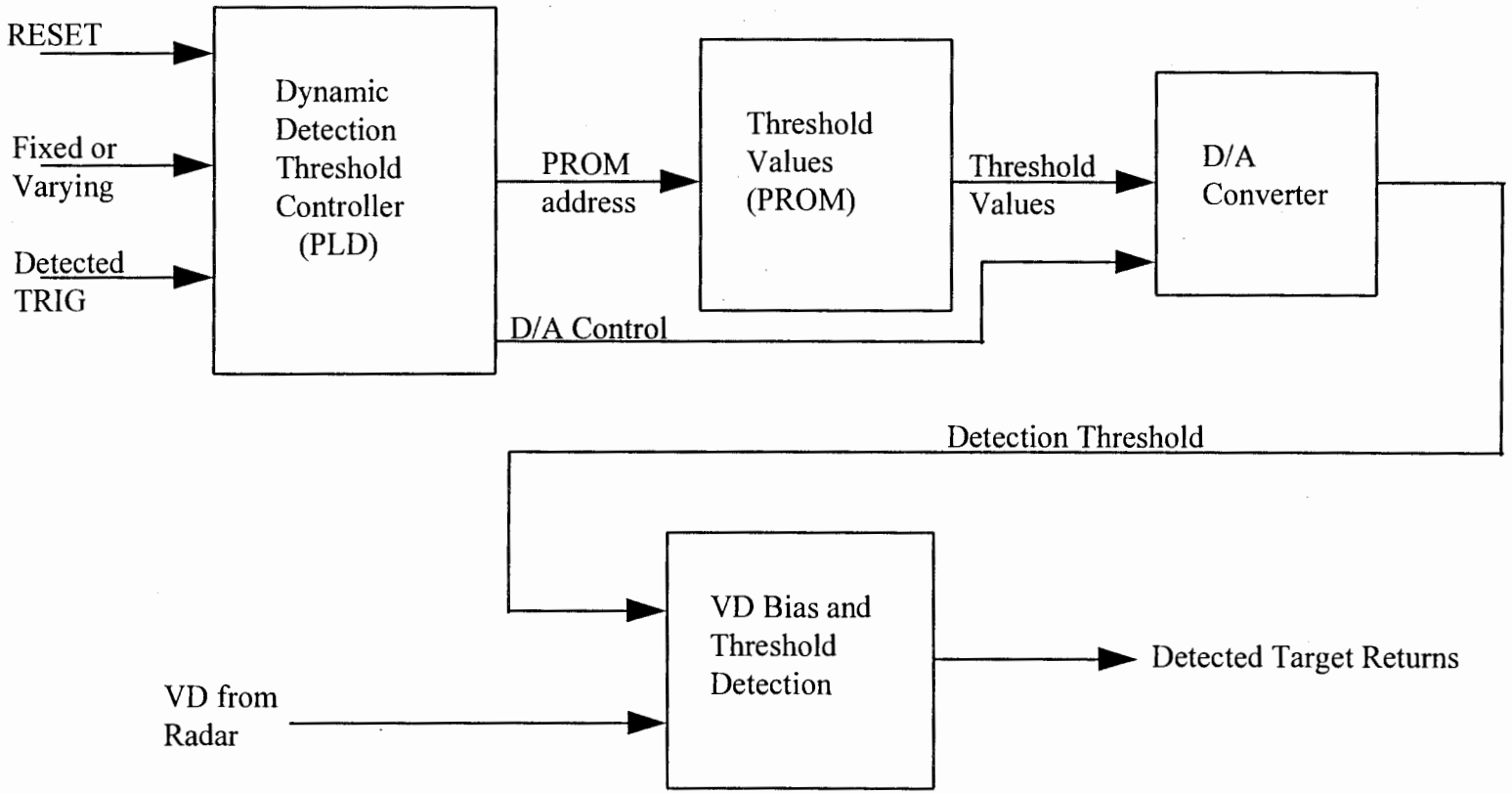


Figure 4.7 Target detection threshold controller block diagram.

controller also issues a latch signal to the D/A converter. This latch signal is output for the first 512 μs after each TRIG. At that point, the value of the threshold remains fixed until the next trigger. Figure 4.8 illustrates the threshold value changes that occur for this algorithm as it is currently programmed. All of the 2048 addresses of the PROM contain a valid threshold value. These values are generated by a program written in C and shown in Appendix E. For this application with 1.33 ms between triggers (750 Hz PRF), only 667 address of the 2048 available in the PROM are accessed. It should be noted that the code for the PROM can easily be modified to adapt to any desired threshold algorithm.

4.3.3 Range Bin Controller

The original LSRC design, because it did not consider the range of a target, was susceptible to the false detection of targets due to noise and other radar signals. The LSRC design incorporates a range bin algorithm in an effort to curtail false detection. Range information can be extracted from a received radar return based upon the delay from the time it was initially transmitted. With a radar signal traveling at or near the speed of light, the range of a target is 150 meters for every 1 microsecond of delay time.

A range bin is a quantization of the range values for detected targets. For example, if a range bin size of 100 meters is chosen, targets found between 0 and 99 meters are placed in the first bin, while those between 100 and 199 meters are placed in the second bin and so on. The quantization level chosen for an application depends upon the range accuracy that is required.

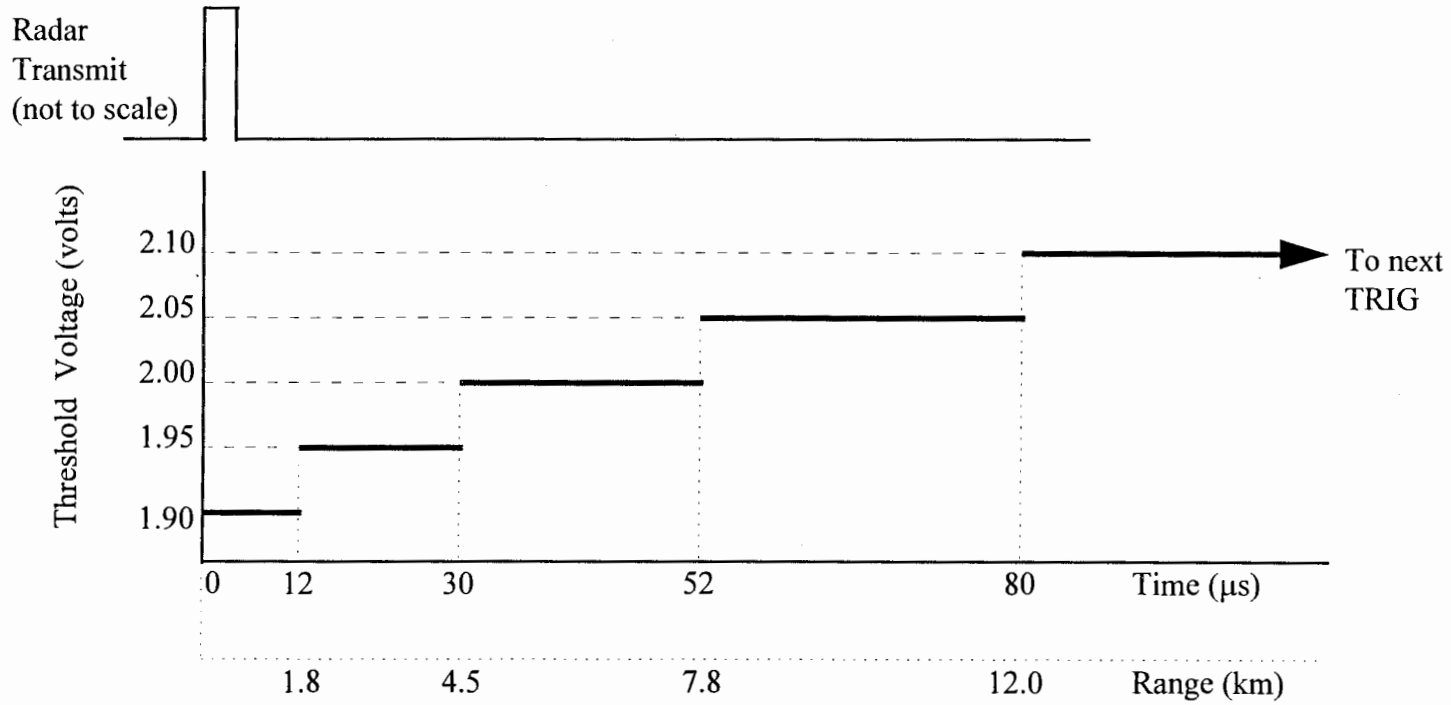


Figure 4.8 Detection threshold values relative to radar pulse transmission.

For this application, a simple range bin algorithm was desired to make the LSRC less susceptible to detection of false targets and still maintain all of the detection capability of the original design. A bin size of 150 meters was selected corresponding to a delay time of 1 μs per bin. Targets are required to be in a given range bin for at least two consecutive radar transmissions to be considered a target. To avoid any detection problems caused by a target directly on a range bin boundary, two overlapping search windows are implemented. The windows overlap in time by 500 ns and the radar return signal is sampled for targets at 125 ns intervals. If a target is found in either search window, the corresponding bin is marked as containing a target. If a target is within 500 ns of the edge of either search range, the bin for both windows is marked as containing a target. A functional timing diagram in Figure 4.9 illustrates the overlapping search ranges and range bins algorithm. It can be seen from Figure 4.9 that if a target falls in the middle of a search range, only one range bin is marked with a target as in the case of targets A and C. Targets B and D fall 500 nanoseconds from the edge of a range bin, causing two range bins to be marked as containing the targets. In this manner all boundary conditions due to range quantization are considered. For a real target to move from one range bin to the next in one transmission of the radar would require it to change its range relative to the radar by 75 meters in 1.33 milliseconds or 202500 kilometers per hour.

The circuit is implemented with a PLD and a 4k x 4 static random access memory (SRAM). A blanking function as described in Section 4.1 is implemented as part of the range bin algorithm. Eight control bits are accepted by the circuit from the computer interface. Each bit carries a weight of 125 ns for a maximum blanking interval of 32 μs.

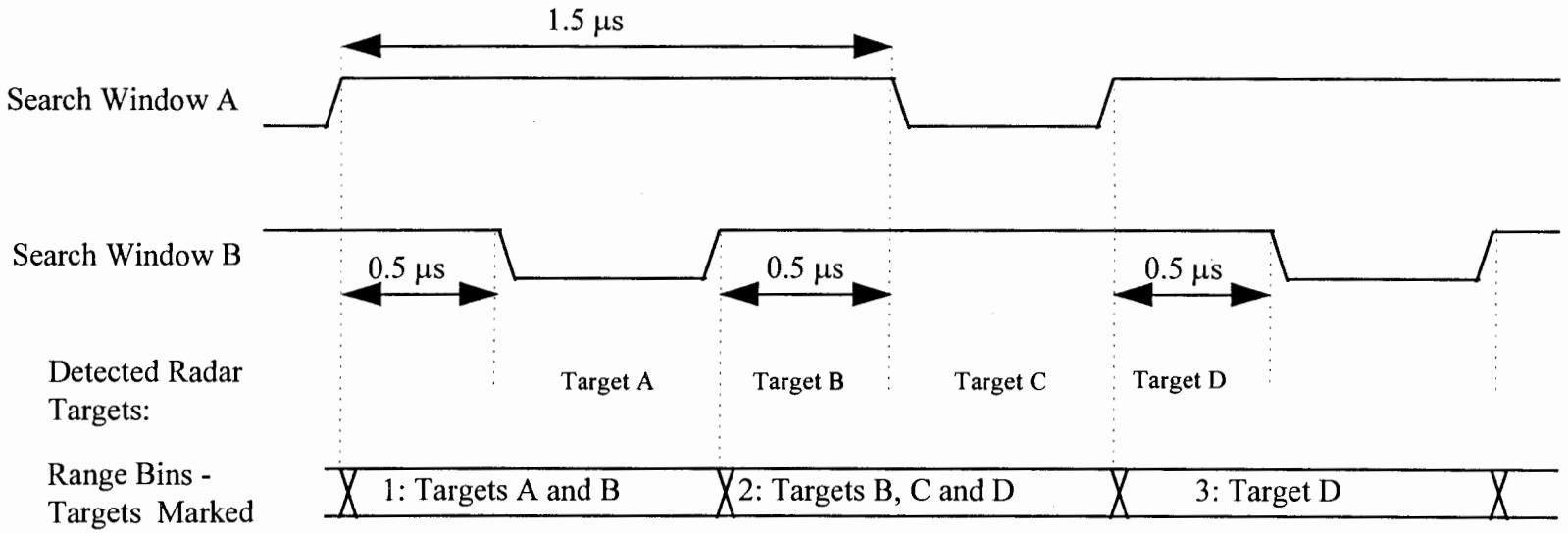


Figure 4.9 LSRC target search and range bin algorithm.

Radar returns are not detected as targets during the blanking interval. Because time is measured in the LSRC with respect to the initial falling edge of the radar's TRIG signal, the blanking must be set to at least 8 μs to blank out the transmission feedback pulse shown in Figure 4.3 for the radar operating at the 750 Hz PRF rate.

At power up or reset of the circuit, the range bin controller PLD writes logic zeroes to all of the memory locations within the SRAM. These locations are then read back into the controller to ensure that all locations are at logic zero. If any logic one levels are found, the laser enable signal (LSRN) is not issued to the laser. Once the SRAM has been checked, the controller begins detecting targets in the overlapping window fashion described previously. The LSRN signal is not issued to the laser until it has been determined that no targets are present in any range bin for three consecutive radar transmissions. As targets are detected, logic one levels are written to the memory locations in the SRAM that represent the range bins. After each radar transmission, each range bin is read to determine if a target was present there previously. The range bin controller will look for a target in a given range bin for either two or three consecutive radar transmissions before disabling the laser. The number of consecutive passes used to determine a valid target is selected by a control bit issued from the computer interface.

When a valid radar target is found, the laser is disabled and the range bin number is made available to the computer interface. This eight bit number represents the number of microseconds delay encountered from the initial falling edge of the radar's TRIG signal to

the time the target is detected. It may be converted to an estimated target range by the following formula,

$$ETR = 150 * (RBN - 5.5), \quad (4.1)$$

where:

ETR = estimated target range (meters),
RBN = range bin number.

The -5.5 μs in the formula is derived from the fact that the actual radar transmission occurs approximately 6.5 μs after the TRIG pulse and that the LSRC requires 1 μs to detect it.

4.3.4 Built-in-Test Functions

Several built-in-tests are performed on the LSRC. In the event of a failure of any of these BIT functions, an error condition is generated, causing the laser enable signal to be changed from an asserted to an de-asserted state. Worst case timing analysis shows that the laser enable signal is changed from an asserted to an de-asserted state in less than 2 microseconds for either a detected hardware failure or a detected target.

The time delay between the radar's trigger signals is continuously monitored. If more than 2.56 ms elapses between triggers, an error is flagged and a radar malfunction error is reported via the computer interface. The time allowed before a radar error is reported is 25 percent more than any currently known PRF rate used by the R72 radar. The main purpose of this function is to detect a failure of the radar and it is not intended to flag minor inconsistencies in the radar's PRF rate. A radar error is also generated if the radar is operated

at the 2000 Hz PRF rate. This mode was intended only for short range operation and the 80 ns pulses are not detectable by the LSRC.

One PLD on the LSRC acts as the central built-in-test controller for the monitoring function used to detect device failures. This device periodically issues signals to four other PLD's on the LSRC. The signal is toggled from a logic high to a logic low and back again. If the expected response is not received, an error is flagged and the device causing the error is reported to via the computer interface and the on card display.

Another feature of the LSRC is the on board seven segment display which reflects encoded status messages and is driven by the built-in-test controller. The status of the circuit may be decoded as shown in Table 4.1.

Table 4.1 LSRC encoded display interpretation.

Displayed Value	Interpretation
0.	Waiting for first trigger from radar
0	LSRC OK
1.	Shutdown for target
2.	Shutdown for radar malfunction
3.	Shutdown for failure of SRAM test
A.	Shutdown, suspect failure of STD Controller device
b.	Shutdown, suspect failure of Clock Divider device
c.	Shutdown, suspect failure of Stagger Controller device
d.	Shutdown, suspect failure of Range Bin Controller device
8.	Shutdown, BIT failure - cause unknown

4.3.5 STD Interface Controller

The LSRC is designed with an STD bus interface to allow it to be controlled and monitored via an embedded controller or microprocessor. Control bits are used to reset the

LSRC, set up the amount of desired pulse stagger for the transmitted PRF, turn the dynamic radar detection threshold algorithm on and off, set up the desired amount of blanking and to set the number of consecutive times that a target must be found in a given range bin to classify it as a target. Valid ranges and values for these parameters as well as a description of the available status information is contained in the LSRC Interconnect Diagram shown in Appendix C.

Chapter 5

Safety Radar Subsystem Test

5.1 LSRC Bench Testing

A prototype Laser Safety Radar Controller (LSRC) as described in Chapter 4 was constructed by the author in a wire-wrap configuration. The wire wrap board was first tested in a laboratory environment using simulated radar signal inputs. An STD based computer was not initially available. Set up conditions for the LSRC were accomplished using DIP switches designed into the wire wrap card. Proof of design tests were performed in this manner on all of the LSRC functions. Refer to the schematics in Appendix A for connection points discussed for the individual tests.

5.1.1 Stagger Controller Test Results

A TTL level square wave was input to the stagger control circuit and the waveform received is shown in Figure 5.1 (Figure 5.1 is not drawn to scale). With the input applied to J4-2, the output was monitored at J4-5. Table 5.1 shows the measured delay times for each valid value of the stagger control bits.

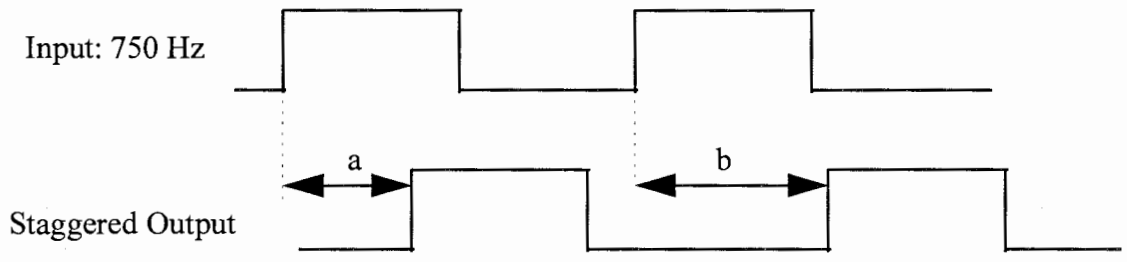


Figure 5.1 Measured delay of staggered output.

The 4 +/- 2 microsecond delay of the signal is induced by the detection of an asynchronous signal by a 500 kHz clock and is expected. The table shows that the circuit operates as expected, adding 0, 2, 4 or 6 microseconds of delay to every other pulse.

Table 5.1 Measured delay values of staggered output.

Value of Stagger Control Bits (MSB to LSB)	Delay Value a (μs)	Delay Value b (μs)
00	4 +/- 2	4 +/- 2
01	4 +/- 2	6 +/- 2
10	4 +/- 2	8 +/- 2
11	4 +/- 2	10 +/- 2

5.1.2 Target Processing Test Results

To simulate trigger and target inputs, two pulse generators were used with the simulated TRIG input applied to J2-5 used as a trigger for a delayed signal simulating the radar target on VD. Adjusting the amount of delay moved the "target" range as required. In this manner, the blanking, detection threshold variance and the range bin algorithm were verified to operate as expected. Simulated targets were input at the ranges shown in Table 5.2 and the LSRC was shown to pull the LSRN signal low indicating a laser shutdown condition was detected. The value of the target range was determined by reading the value of the register using an oscilloscope. Initially, the threshold algorithm was disabled and a fixed detection voltage of 1.95 volts was used. The commanded blanking value was then set to 12 μs with the simulated target at 10 μs and the LSRN signal remained at its high level as expected. The blanking value was then placed at 9 μs and the target was detected.

Table 5.2 LSRC response to simulated radar targets.

Simulated Target Range (μs)	Reported Target Range (μs)	LSRN = Logic Zero (Shutdown)
10	8	YES
40	38	YES
80	78	YES
90	88	YES

The time constant of the detection circuit for the radar TRIG signal induces a delay of approximately 1 μs. This is the main source of the discrepancy between the reported range and simulated range. The range values are intended for reference only and the estimates have an uncertainty of +/- 300 meters.

The above tests were repeated after enabling the thresholding algorithm and the results were re-verified. BIT functions were tested and verified by removing appropriate components to simulate hardware failures.

5.2 Integration of the LSRC with the Raytheon Radar

The next phase of testing of the LSRC required it to interface with the Raytheon R72 radar. A LIDAR research instrument designed at the Pennsylvania State University known as the WAVE-LARS system (LARS) also incorporates a radar safety subsystem using the R72 radar unit and the original version of the LSRC described in Chapter 4. The wire wrap LSRC was installed into this system to finalize the proof-of-design testing. The antenna was placed in an outdoor location pointing at an area with no air traffic for the following initial tests.

5.2.1 Verification of Blanking and Range Bin Functions

The time between the falling edge of the radar's TRIG signal and the end of the transition of the radar's target detection indicator VD caused by the transmission of the radar was measured and verified to be 9 μs when operating the radar at a 750 Hz PRF with a 700 ns transmission pulse. Refer to Figure 4.3. With the blanking value commanded to be less than 9 μs, the status display for the LSRC showed it to be in a shutdown state due to a target. Note that the LSRN signal was tested to ensure that is was in the de-asserted state when the shutdown status was reported. This same condition was tested for both available range bin algorithms and was found to operate as expected. The blanking value was then set to 9.5 μs and it was verified that no shutdown conditions occurred.

5.2.2 Verification of Pulse Stagger Function

The pulse stagger function was enabled and the tests described in Section 5.2.1 were executed to verify that altering the transmissions of the radar did not affect operation. All four settings for this function were tested in this manner and the subsystem operated as expected.

5.2.3 Verification of Target Detection Threshold Function

The target detection threshold function was enabled and the tests described in Section 5.2.1 were again executed to verify that the operation of the subsystem was not affected.

5.3 SRS Testing Summary

Tests conducted by the author on the Safety Radar subsystem involved pointing the radar antenna at fixed targets and monitoring the status display on the LSRC to determine the state of the hardware. Trees and mountains at ranges from less than one kilometer and up to 11 kilometers caused the subsystem to immediately enter a laser shutdown state. Estimated range values read from the LSRC correlated with the range to target values shown on the radar display. A final set of tests were conducted which involved pointing the radar antenna at moving aircraft. Approximately ten trials were run on small to medium aircraft whose distances from the radar ranged from 4 to 10 kilometers. The Safety Radar subsystem was shown to enter its shutdown state in all trials run.

The Raytheon R72 radar has been extensively tested for its use in this application. Data taken by Jim Yurak using the R72 in the LARS safety subsystem is shown in Figures 5.2, 5.3 and 5.4. These figures show the response of the radar's VD signal when detecting aircraft in three separate trials [13]. In addition, field testing of the subsystem with the original LSRC was accomplished at the Point Mugu Naval Air Station in Point Mugu, California. During these tests, a Navy A-7 Corsair II aircraft performed 46 fly-bys at

different altitudes and orientations. The subsystem successfully detected the aircraft and invoked a shutdown of the laser for each trial [13,16].

A printed circuit version of the LSRC is currently in the process of integration into the LAPS system by the Applied Research Laboratory (ARL). The documentation given in the Appendix reflects the design as it exists to this point. Baselines for the wire wrap and printed circuit designs are currently being maintained by ARL.

Further field testing of the Safety Radar subsystem with the LSRC hardware is necessary to prove its safety and reliability. Tests to verify the maximum and minimum ranges of the subsystem should be executed with the hardware in both two pulse and three pulse integration modes. The subsystem should be tested at both the 1500 Hz and 750 Hz PRF rates to determine the optimal operating condition. Finally, the subsystem should be operated in areas where other marine radars are transmitting with its pulse staggering engaged to verify proper operation.

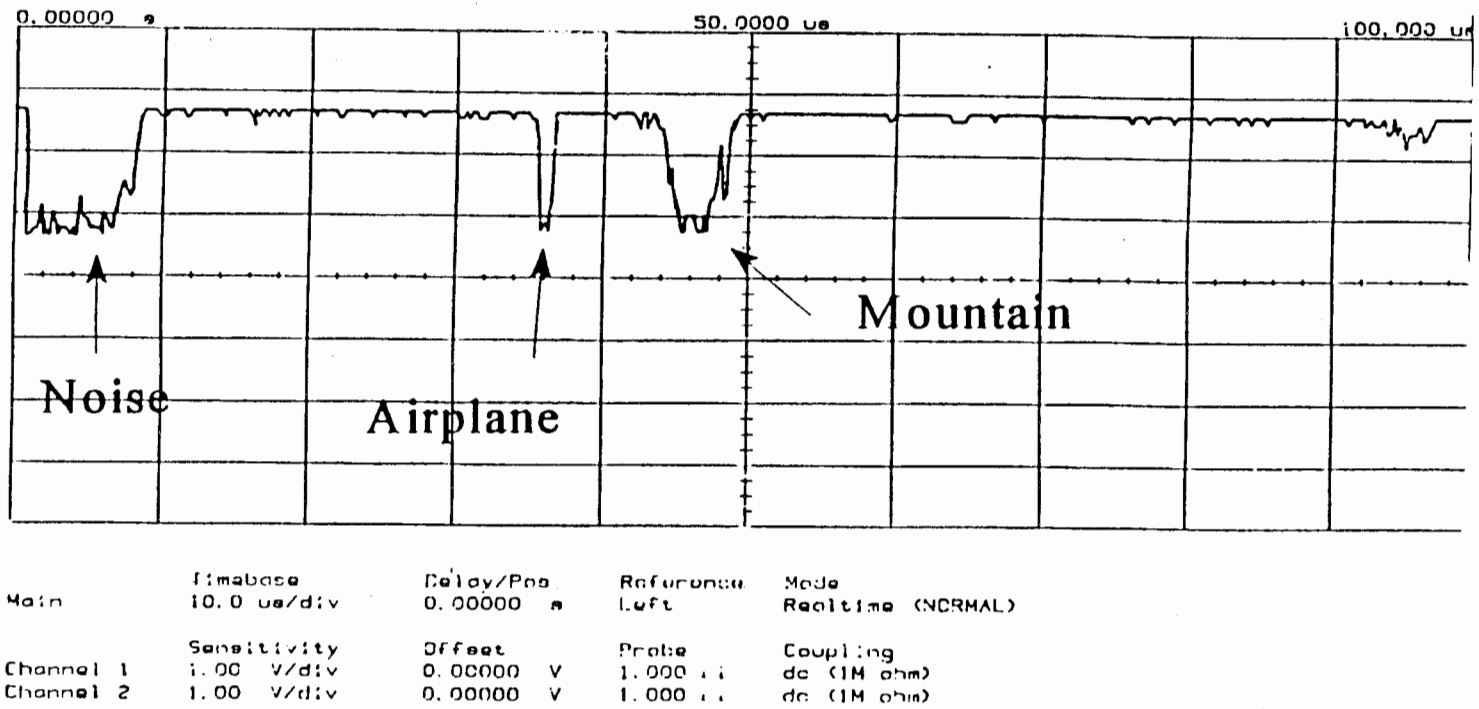


Figure 5.2 Time plot of R72 radar detection of aircraft at 5.3 km range [13].

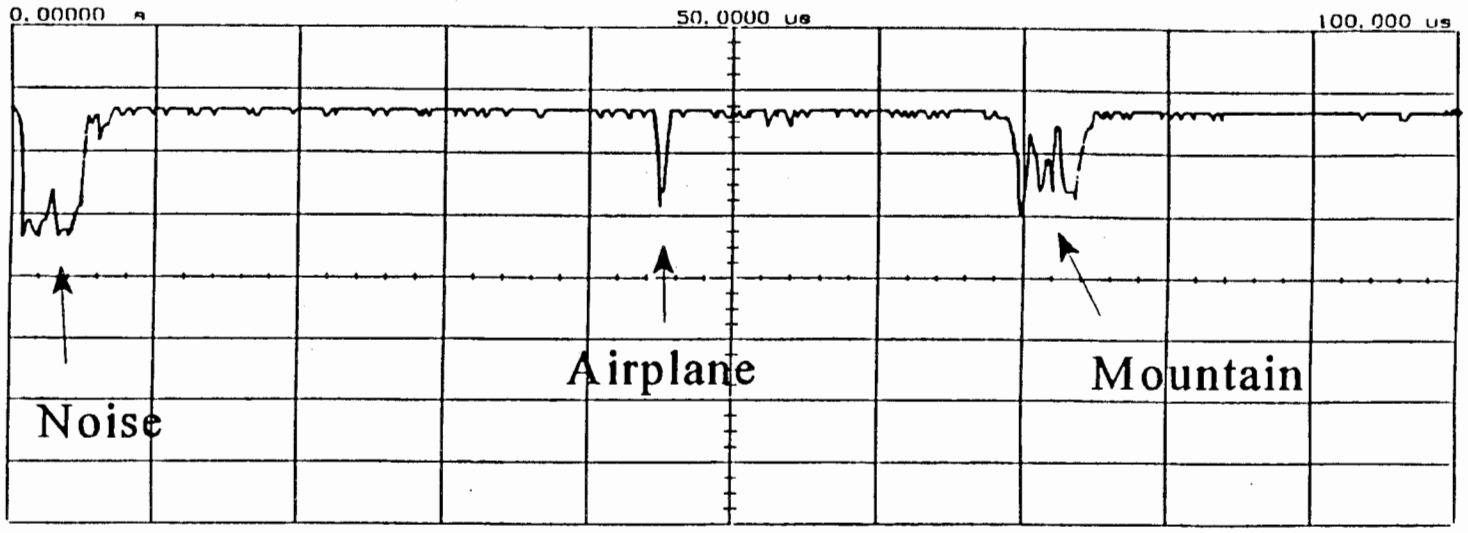


Figure 5.3 Time plot of R72 radar detection of aircraft at 6.8 km range [13].

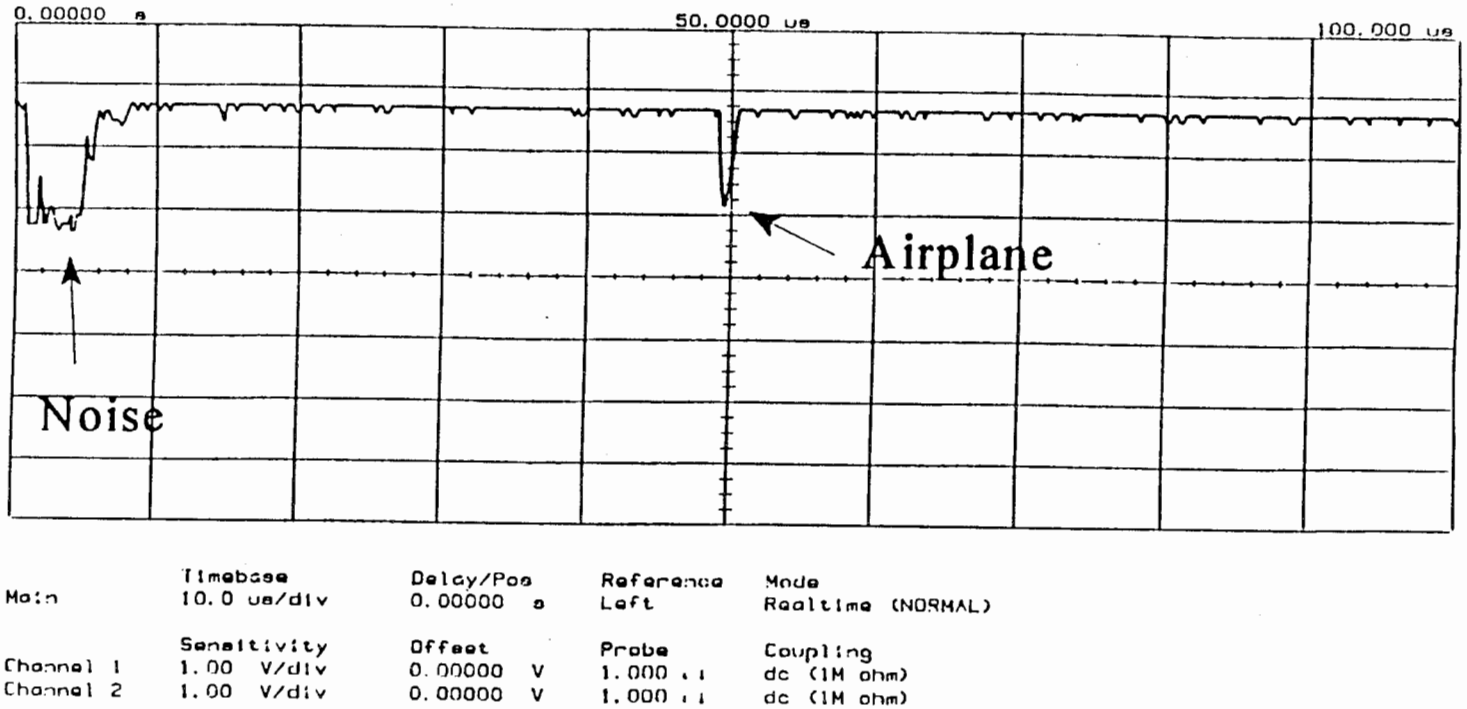


Figure 5.4 Time plot of R72 radar detection of aircraft at 7.4 km range [13].

Chapter 6

Conclusions

6.1 Summary of Features

The Safety Radar subsystem offers a valid solution to the problem of protecting operating aircraft from the hazard presented by the use of an atmospheric LIDAR system. Through the use of a commercially available marine X-band radar, the subsystem is cost effective, reliable and easily maintained. The addition of a parabolic dish antenna in place of the delivered sweep antenna yields a 5.6 degree protection zone around the laser. The radar provides more than adequate coverage for this area up to the maximum flight ceilings for commercial aircraft. The LSRC hardware interfaced to the radar provides several key features for the subsystem. Several self monitoring functions are provided for the hardware and radar to ensure safe failure modes in the event of a detectable error or malfunction. Functions added to increase the reliability of the subsystem include the implementation of range gating and pulse staggering. Each of these features should reduce false shutdown conditions caused by noise and the operation of other X-band radars. The computer interface allows for a flexible subsystem that can easily be adapted to changing operational requirements without modifying the hardware.

Caution is advised in the following areas: 1) applications of a Safety Radar subsystem as described in this thesis that bypass or inhibit any of its safety features must be carefully analyzed to insure that all possible failure modes of the modified system have been considered; 2) no components should be added between the Safety Radar subsystem and the laser Q-switch inhibit without careful re-evaluation of all safety issues and extensive analysis.

6.2 Operational Considerations

It is recommended for maximum safety that the Safety Radar subsystem be operated only in areas where air traffic is encountered in level flight or in shallow angles of ascent or descent to ensure a maximum radar cross-section is available to the radar. These aircraft should also be operating at altitudes above 300 meters.

Operation in rain or snow conditions at precipitation rates of more than 25 mm/hr (1 in/hr) is expected to degrade the maximum detection range performance.

The Safety Radar subsystem contains a computer interface such that it can be easily controlled via an embedded controller or microprocessor on the STD bus. The status of the radar and the LSRC hardware is continuously available to the computer. The hardware, once it enters a shutdown state will not re-enable the laser until such time as a reset is commanded by the computer. The computer program is thus expected to handle these shutdown conditions and determine when it is appropriate to reset the subsystem and resume searching for targets. After a reset, the hardware will not enable the laser until three radar transmissions have occurred with no targets detected.

The LSRC hardware is aware of the TRIG pulse sent to it from the radar but has no way of determining if the transmission of the radar pulse actually occurred. One possible solution to this is to construct a radar band power detector that would interface directly to the computer. This device would contain an RF power detector, a passive component that outputs a DC level based upon the RF power level it receives. With a small dipole antenna mounted in the vicinity of the parabolic dish antenna and a feed running to the detector, only simple signal conditioning

techniques would be required to interface it to the computer and determine that radar transmissions were occurring.

For the subsystem to determine the status of the radar receiver, the blanking value should periodically be set back to the point that the radar transmission feedback pulse is in the detection range and verify that it is received as a target.

Calibration of the radar's transmitted power and receiver sensitivity using a radar test set is recommended at approximately six month intervals along with a visual inspection of both the radar and the custom hardware.

Before the Safety Radar subsystem can be used as a completely autonomous instrument, further field testing is required with the use of human observers. Proven performance in this mode of operation will yield the confidence necessary to move toward the goal of an unattended subsystem.

6.3 Suggestions for Future Work

Further design work to enhance the capabilities of the Safety Radar subsystem could be investigated in the following areas: (1) the external detection of the radar transmissions discussed earlier in this chapter; (2) development of software to automatically set up the blanking value for a selected PRF rate; (3) development of a separate instrument to detect targets under 300 meters; (4) design enhancements to the LSRC including using a RAM in place of the EPROM in the variable detection threshold circuit.

References

[1] Schmeisser, E. T., Laser exposure effects on visual functions: measurements by electrophysiology - a selective review, *Laser Safety, Eyesafe Laser Systems, and Laser Eye Protection*, SPIE Vol. 1207, pp.14-15, January 1990.

[2] Winburn, D. C., *Practical Laser Safety*, Marcel Dekker, Inc., pp. 19-25, 1990.

[3] Philbrick, C. R.,et al., *LAMP TO LAPS: The Transition from a Research to an Operational LIDAR Sensor for Atmospheric Properties,* Vol V, LAMP LIDAR SAFETY PLAN, 30 September 1992.

[4] Philbrick, C. R., et al., *Measurements of the High Latitude Middle Atmosphere Dynamic Structure Using Lidar*, Environmental Research Papers, No. 967, Hanscom AFB, Massachusetts, 18 February, 1987.

[5] *Model R70 Series Raster Scan Radar Systems,* Instruction Manual, Raytheon Marine Company, Hudson, New Hampshire.

[6] Interoffice Correspondence, Electrical Test Report, AS15-90 15 inch Diameter Antenna, Seavey Engineering Associates, Inc., Cohasset, Massachusetts.

[7] *1990-1991 Jane's Aircraft*, Marston and company, ltd.

[8] Skolnik, Merrill I., *Introduction to Radar Systems*, McGraw-Hill Book Company, New York, New York, 1980.

[9] Blake, Lamont V., *Radar Range-Performance Analysis*, Lexington Books, Boston, Massachusetts, 1980.

[10] Knott, Eugene F., Shaeffer, John F., Tuley, Michael T., *Radar Cross Section*, Artech House, Norwood, Massachusetts, 1993.

[11] Stimson, George W., *Introduction to Airborne Radar*, Hughes Aircraft Company, El Segundo, California, 1983.

[12] Wiley, Richard G., *Electronic Intelligence: The Analysis of Radar Signals*, Artech House, Norwood, Massachusetts, 1982.

[13] Yurak, J., *Design of a Safety Radar System to Enable Safe Off Zenith Laser Transmission*, Master of Science Thesis, The Pennsylvania State University, 1994.

[14] Moyer, William W., Interoffice Correspondence, *Preliminary Design of LAMP Radar Interface*, SE93-160, Applied Research Laboratory, University Park, Pennsylvania, 1993.

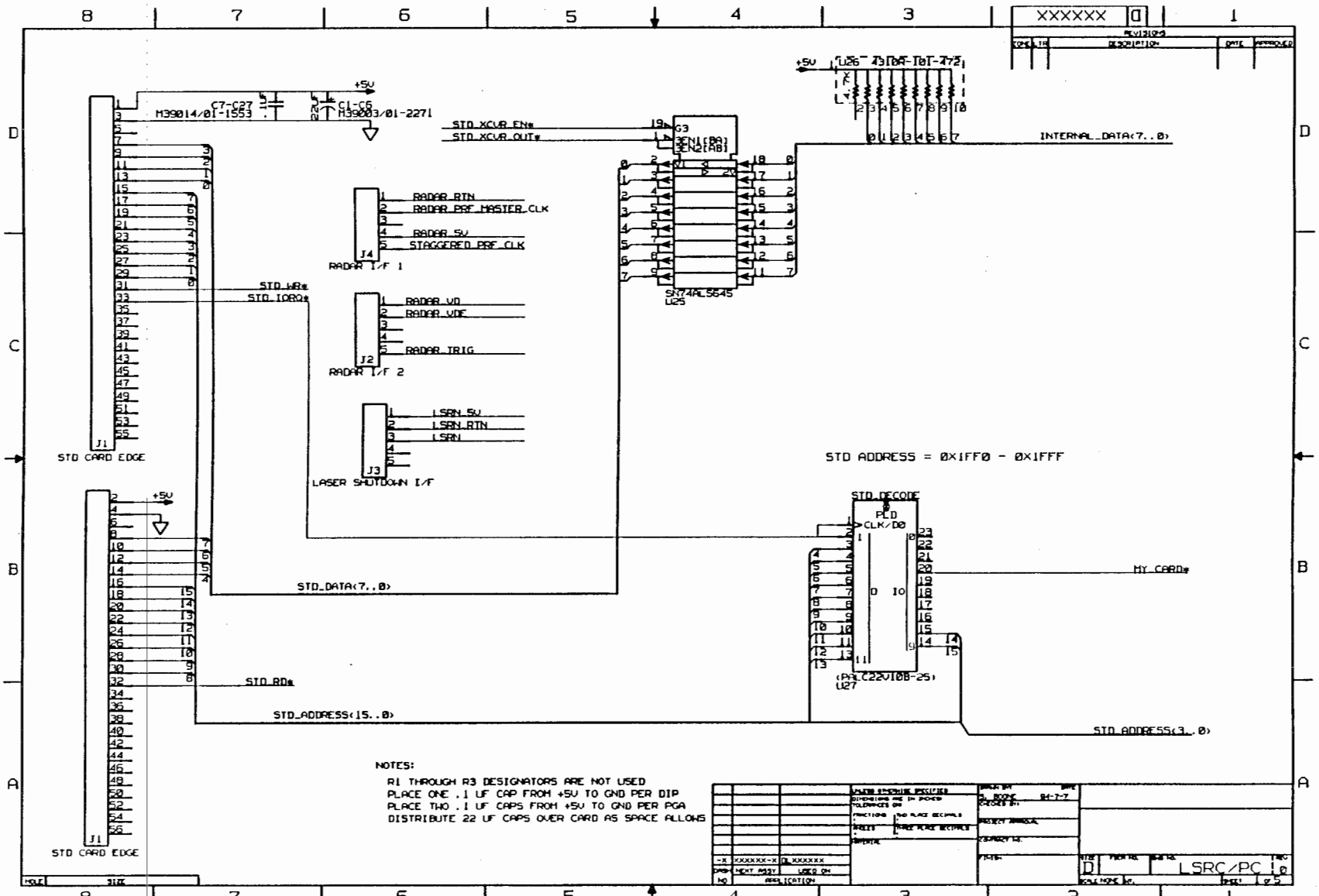
[15] Moyer, William W., Interoffice Correspondence, *Design of LAMP Radar Safety Subsystem*, SE93-200, Applied Research Laboratory, University Park, Pennsylvania, 1993.

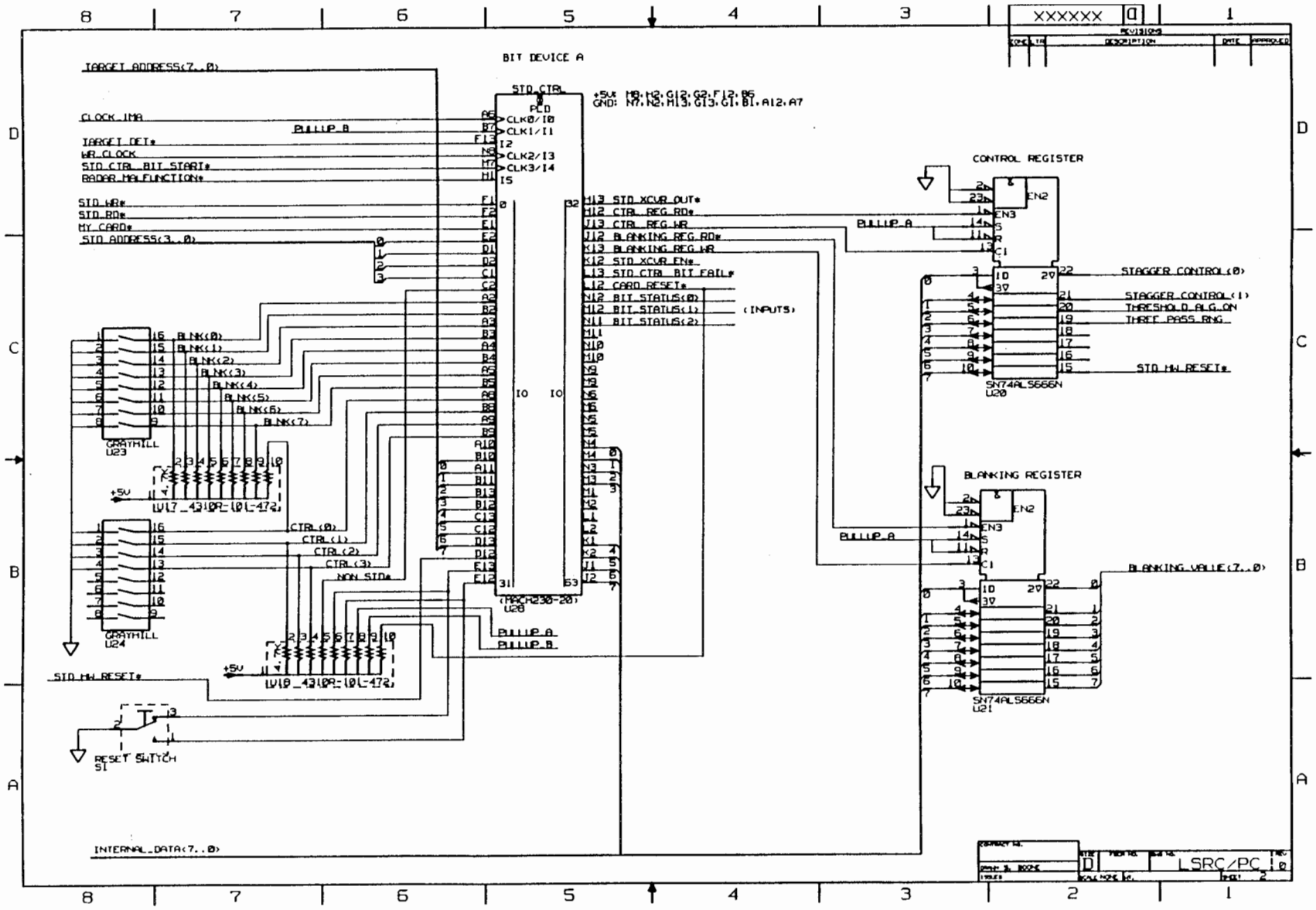
[16] Interoffice Correspondence, *LAMP Lidar Tests at Point Mugu: Final Report,* Applied Research Laboratory, University Park, Pennsylvania, 15 June 1994.

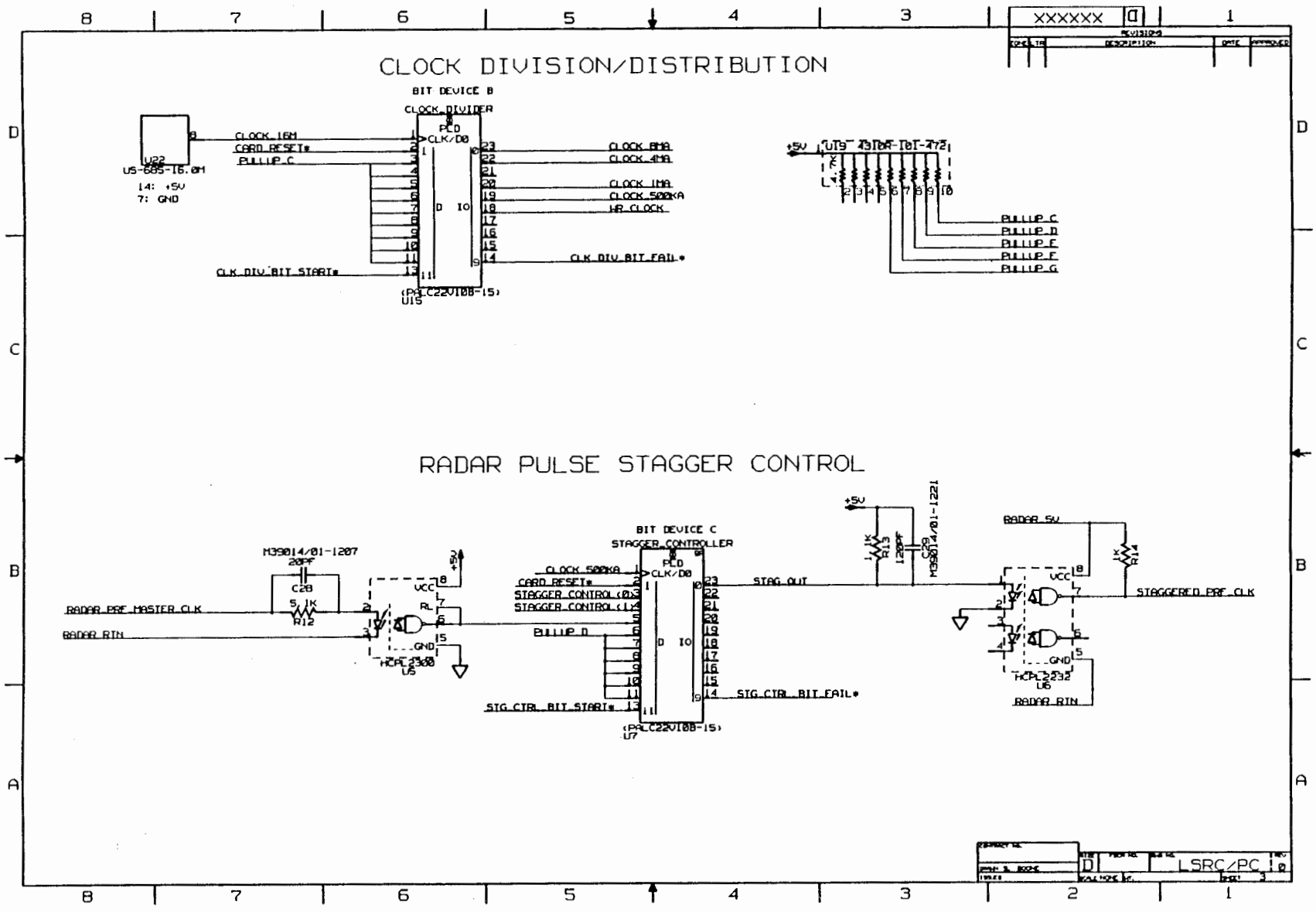
Appendix A

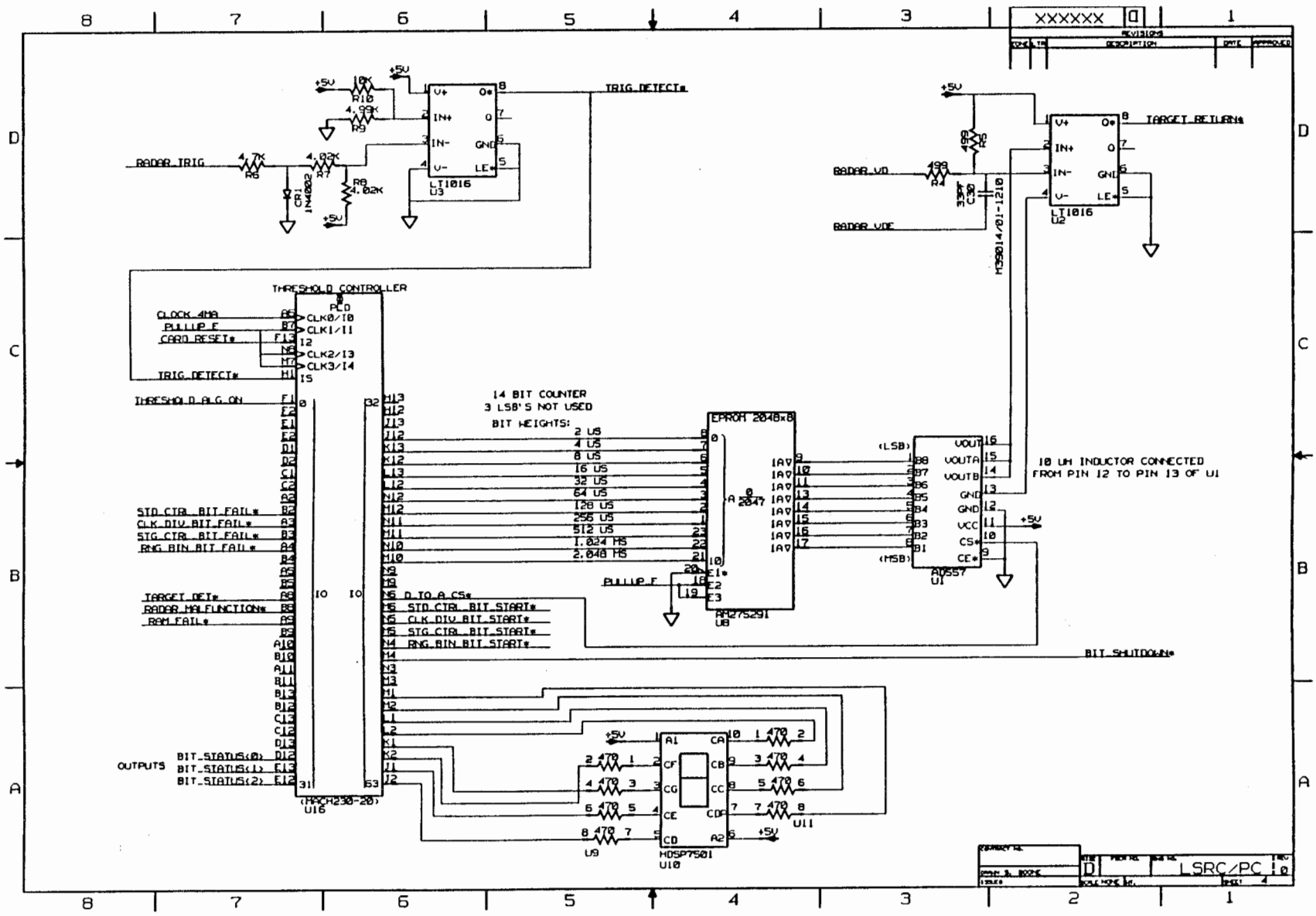
LSRC Schematics

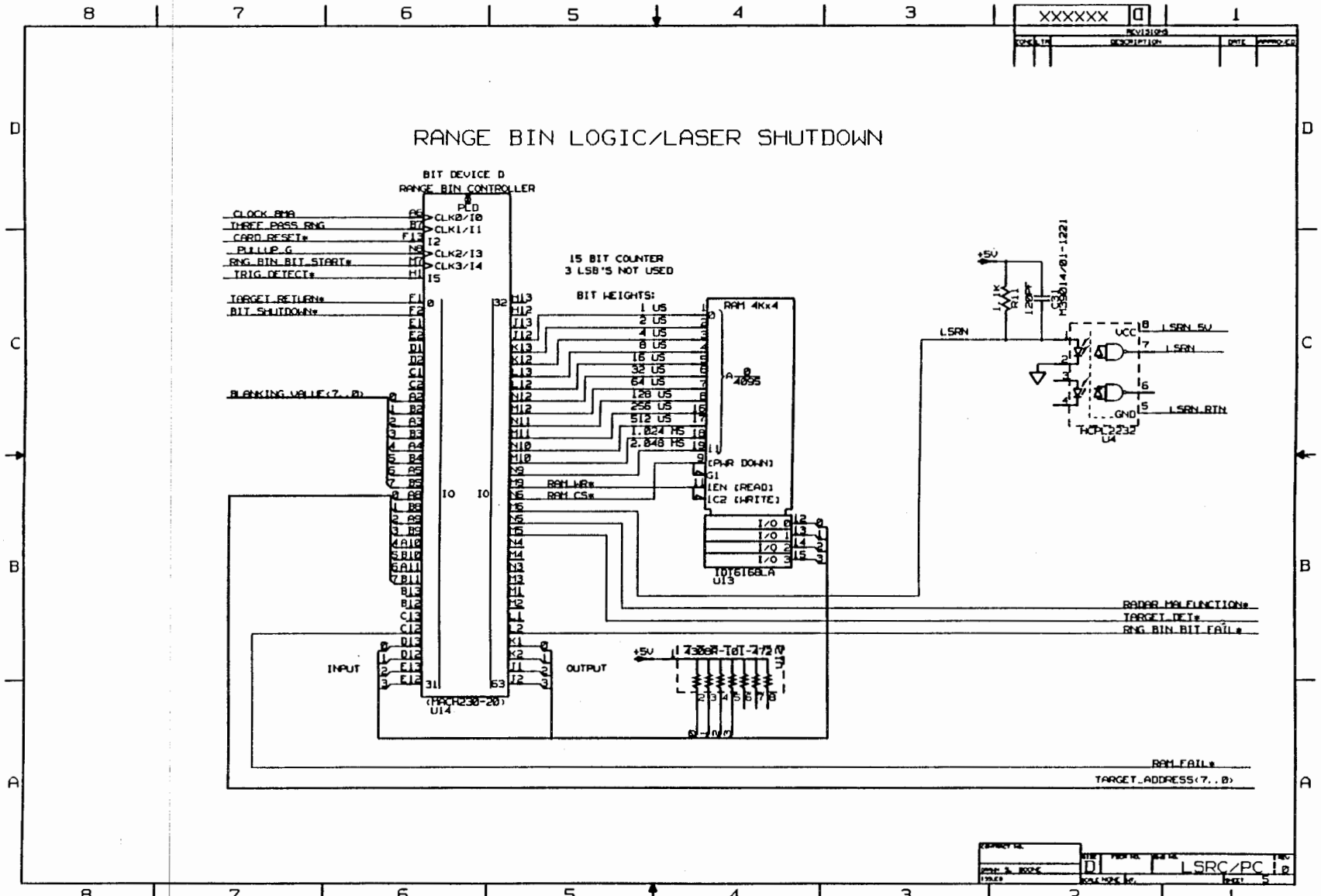
The LSRC schematics are contained on pages 62 - 66.


XXXXXX
+5V
C7-C27
M39014/01-1553
C1-C6
M39003/01-2271
STD_XCVR_EN*
STD_XCVR_OUT*
SN74AL5645
U25
INTERNAL_DATA<7..0>
RADAR_RTN
RADAR_PRF_MASTER_CLK
RADAR_SV
STAGGERED_PRF_CLK
J4
RADAR I/F 1
STD_WR*
STD_IORQ*
RADAR_VD
RADAR_VDE
RADAR_TRIG
J2
RADAR I/F 2
LSRN_SV
LSRN_RTN
LSRN
J3
LASER SHUTDOWN I/F
J1
STD CARD EDGE
STD ADDRESS = 0X1FF0 - 0X1FFF
STD_DECODE
PLD
CLK/D0
MY_CARD*
(PALC22V10B-25)
U27
STD_DATA<7..0>
STD_RD*
STD_ADDRESS<15..0>
STD_ADDRESS<3..0>
NOTES:
R1 THROUGH R3 DESIGNATORS ARE NOT USED
PLACE ONE .1 UF CAP FROM +5V TO GND PER DIP
PLACE TWO .1 UF CAPS FROM +5V TO GND PER PGA
DISTRIBUTE 22 UF CAPS OVER CARD AS SPACE ALLOWS
LSRC/PC


BIT DEVICE A
XXXXXX
REVISIONS
ZONE
LTR
DESCRIPTION
DATE
APPROVED
TARGET ADDRESS(7..0)
CLOCK_1MA
PULLUP_B
TARGET_DET*
WR_CLOCK
STD_CTRL_BIT_START*
RADAR_MALFUNCTION*
STD_WR*
STD_RD*
MY_CARD*
STD_ADDRESS(3..0)
STD_CTRL
PLD
CLK0/I0
CLK1/I1
CLK2/I3
CLK3/I4
+5V
GND
STD_XCVR_OUT*
CTRL_REG_RD*
CTRL_REG_WR
BLANKING_REG_RD*
BLANKING_REG_WR
STD_XCVR_EN*
STD_CTRL_BIT_FAIL*
CARD_RESET*
BIT_STATUS(0)
BIT_STATUS(1)
BIT_STATUS(2)
(INPUTS)
(MACH230-20)
U20
CONTROL REGISTER
EN2
EN3
SN74ALS666N
U20
STAGGER_CONTROL(0)
STAGGER_CONTROL(1)
THRESHOLD_ALG_ON
THREE_PASS_RNG
STD_HW_RESET*
BLANK(0)
BLANK(1)
BLANK(2)
BLANK(3)
BLANK(4)
BLANK(5)
BLANK(6)
BLANK(7)
GRAYHILL
U23
LV17_430R-10L-472
CTRL(0)
CTRL(1)
CTRL(2)
CTRL(3)
NON_STD*
GRAYHILL
U24
LV10_430R-10L-472
BLANKING REGISTER
BLANKING_VALUE(7..0)
SN74ALS666N
U21
RESET SWITCH
S1
INTERNAL_DATA(7..0)
LSRC/PC
SHEET 2


CLOCK DIVISION/DISTRIBUTION
BIT DEVICE B
CLOCK_DIVIDER
CLOCK_16M
CARD_RESET*
PULLUP_C
CLOCK_8MA
CLOCK_4MA
CLOCK_1MA
CLOCK_500KA
HR_CLOCK
CLK_DIV_BIT_FAIL*
CLK_DIV_BIT_START*
PULLUP_C
PULLUP_D
PULLUP_E
PULLUP_F
PULLUP_G
RADAR PULSE STAGGER CONTROL
BIT DEVICE C
STAGGER_CONTROLLER
RADAR_PRF_MASTER_CLK
RADAR_RTN
STAG_OUT
STAGGERED_PRF_CLK
STIG_CTRL_BIT_START*
STIG_CTRL_BIT_FAIL*
LSRC/PC


XXXXXX
RADAR_TRIG
4.7K
R6
4.02K
R7
4.02K
R8
CR1
1N4002
LT1016
U3
TRIG_DETECT*
+5V
10K
R10
4.99K
R9
RADAR_VD
RADAR_VDE
499
R5
499
R4
.39UF
C30
LT1016
U2
TARGET_RETURN*
M39014/01-1210
THRESHOLD CONTROLLER
PCD
CLOCK_4MA
PULLUP_F
CARD_RESET*
TRIG_DETECT*
THRESHOLD_ALG_ON
CLK0/I0
CLK1/I1
CLK2/I3
CLK3/I4
STD_CTRL_BIT_FAIL*
CLK_DIV_BIT_FAIL*
STG_CTRL_BIT_FAIL*
RNG_BIN_BIT_FAIL*
TARGET_DET*
RADAR_MALFUNCTION*
RAM_FAIL*
OUTPUTS
BIT_STATUS(0)
BIT_STATUS(1)
BIT_STATUS(2)
(PACM230-20)
U16
14 BIT COUNTER
3 LSB'S NOT USED
BIT HEIGHTS:
2 US
4 US
8 US
16 US
32 US
64 US
128 US
256 US
512 US
1.024 MS
2.048 MS
EPROM 2048x8
AM27S291
U8
D_TO_A_CS*
STD_CTRL_BIT_START*
CLK_DIV_BIT_START*
STG_CTRL_BIT_START*
RNG_BIN_BIT_START*
PULLUP_F
(LSB)
(MSB)
AD557
U1
VOUT
VOUTA
VOUTB
GND
VCC
CS*
CE*
10 UH INDUCTOR CONNECTED
FROM PIN 12 TO PIN 13 OF U1
BIT_SHUTDOWN*
470
U9
MDSP7501
U10
U11
LSRC/PC
D


RANGE BIN LOGIC/LASER SHUTDOWN
BIT DEVICE D
RANGE BIN CONTROLLER
CLOCK_8MA
TMRF_PASS_RNG
CARD_RESET*
PULLUP_G
RNG_BIN_BIT_START*
TRIG_DETECT*
TARGET_RETURN*
BIT_SHUTDOWN*
BLANKING_VALUE(7..0)
15 BIT COUNTER
3 LSB'S NOT USED
BIT WEIGHTS:
1 US
2 US
4 US
8 US
16 US
32 US
64 US
128 US
256 US
512 US
1.024 MS
2.048 MS
RAM 4Kx4
IDT6168LA
U13
RAM_WR*
RAM_CS*
INPUT
OUTPUT
+5V
LSRN
LSRN_5V
LSRN
LSRN_RTN
GND
VCC
RADAR_MALFUNCTION*
TARGET_DET*
RNG_BIN_BIT_FAIL*
RAM_FAIL*
TARGET_ADDRESS(7..0)
LSRC/PC

Appendix B

LSRC Parts List

Quantity	Reference Designator	Part Number	Manufacturer	Description
3	J2,J3,J4	WM4303-ND	Digi-Key	Circuit header .1 right angle friction lock
1	U1	AD557JN	Analog Devices	8 bit D to A - voltage output
2	U2,U3	LT1016CN8	Linear Technologies	High speed comparator
2	U4,U6	HCPL2232	HP	Optical Isolator
1	U5	HCPL2300	HP	Low current optical isolator
3	U7,U15,U27	PALC22V10L-25WC	Cypress	PLD
1	U8	AM27C291-25PC	AMD	2Kx8 CMOS PROM
2	U9, U11	4308R-102-471	Bourns	470 ohm discrete 8 pin sip
1	U10	HDSP7501	HP	7 segment display - common anode
1	U12	4308R-101-472	Bourns	4.7 Kohm pullup 8 pin sip
1	U13	IDT6168LA25P	IDT	4k x 4 CMOS static RAM
3	U14,U16,U28	MACH230-20JC	AMD	PLD - 84 pin PLCC
3	U14,U16,U28	PLCC 84P-T-2	Mackenzie	PLCC to PGA 84 pin sockets
4	U17-U19,U26	4310R-101-472	Bourns	4.7 Kohm pullup 10 pin sip
2	U20,U21	SN74ALS666N	TI	8 bit D-type read back latch
1	U22	US-685-16.0	US Crystal	Crystal oscillator - 16.0 MHz
2	U23,U24	78B08	Grayhill	8 position DIP switch
1	U25	SN74ALS645N	TI	Octal transceiver
1	S1	E112-S-D1-A-B-E	C+K	Momentary contact switch with common
1	L1	54-140-1	Ferronics Inc.	Inductor - 10 uH
1	CR1	1N4002	(Multiple)	Diode
6	C1-C6	M39003/01-2271		Capacitor 22 uf - 15 volt
21	C7-C27	M39014/01-1553		Capacitor 0.1 uf
1	C28	M39014/01-1207		Capacitor 20 pf
2	C29,C31	M39014/01-1221		Capacitor - 120 pf
1	C30	M39014/01-1210		Capacitor - 33 pf
2	R4,R5			Resistor - 499 ohm - 1/8 W
1	R6			Resistor - 4.7 Kohm - 1/8 W
2	R7, R8			Resistor - 4.02 Kohm - 1/8 W
1	R9			Resistor - 4.99 Kohm - 1/8 W
1	R10			Resistor - 10 Kohm - 1/8 W
2	R11,R13			Resistor - 1.1 Kohm - 1/8 W
1	R12			Resistor - 5.1 Kohm - 1/8 W
1	R14			Resistor - 1 Kohm - 1/8 W

Appendix C

LSRC Hardware - Software Interconnect Diagram

```
/* LRSC_ICD.h : LSRC Interconnect diagram
                Created 1/10/95
                Scott P. Boone
                MSEE thesis project


This file will compile in C and should be included in any C program to set
up functions to drive the LSRC STD card.

    The LSRC STD card contains four registers:

 Base Address   Register Name            Register Type
   0x1000           Control Register         Read/Write
   0x1001           Status Register          Read only
   0x1002           Blanking Register        Read/Write
   0x1003           Target Range Register    Read only

 A description of each register and the bit weights follows:
*****************************************************

 Control Register:
 Bit:    |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
         | RST*|  X  |  X  |  X  |3PASS|THRON|STCT1|STCT0|

RST*
Invokes reset of LSRC functions.
  0 = Reset active
  1 = Reset inactive

Note that this reset does not clear the Control or Blanking registers. This
bit must be set to a 1 after power up.

3PASS
  0 = Range bin algorithm looks for a target to be in a range bin for two
consecutive passes before invoking a shutdown condition.
  1 = Range bin algorithm looks for a target to be in a range bin for three
consecutive passes before invoking a shutdown condition.

THRON
    0 = Thresholding algorithm is not used. The threshold value used to
compare the radar input signal for targets remains fixed at approx 1.95v.
    1 = Thresholding algorithm is used. The threshold value used to
compare to the radar input signal varies in time such that it is more
sensitive to longer range targets.
```

STCT1..STCT0

These bits control the amount of delay placed on every other trigger pulse issued to the radar. This is used to give the radar controlled by the LSRC a unique pattern that is less likely to be interferred with by other radars. The delay placed on every other pulse can be derived from the following table:

MSB	LSB	Delay (Microseconds)
0	0	0
0	1	2
1	0	4
1	1	6

The recommend default value of this register after a reset is: 0x83

****

Status Register:

Bit:	7	6	5	4	3	2	1	0
	X	X	NTRG	BIT2	BIT1	BIT0	MFTN*	TARG*

NTRG

1 Indicates that the card has received its first trigger

0 Indicates that the card has NOT received its first trigger

STD

1 = Card is configured to be used in an STD configuration.

0 = Card is configured to be set up with on board DIP switches.

BIT2..BIT0

These bits are used to determine the "health" of the LSRC. The card monitors several functions and will report faults using these bits as follows:

MSB..LSB	Status
0	Hardware status is OK
1	STD controller hardware failure
2	Clock divider hardware failure
3	Stagger controller hardware failure
4	Range bin controller hardware failure
5	RAM Check failure*
6	RESERVED**
7	RESERVED**

Error codes are latched and the card must be reset to clear error conditions.

* This built in test is run only immediately following a power up or a reset.

** These codes are reserved for future use.

MFTN*

1 = Radar trigger pulses occurring as expected.

0 = Radar trigger pulses absent or erratic.***

```
  TARG*
  1 = No target found.
  0 = Shutdown for target.***

*** Once set to 0, this bit will remain at that state until a reset is
carried out.

************************************************************

 Blanking Register:

 Bit: |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
      | BL7 | BL6 | BL5 | BL4 | BL3 | BL2 | BL1 | BL0 |

BL7..BL0
  Set up the blanking value for the detection of targets.
The first several microseconds of time following a trigger are blanked or
ignored due to the possibility of false targets generated by the radar
immediatley following its signal transmission. Each bit = 125 nanoseconds
(1 8Mhz clock period). The possible range of this register is then 0 to 31
microseconds. Testing has shown that ~9 microseconds (0x48) should be the
minimum setting.

The recommend default value of this register after a reset is: 0x48
*******************************************************

 Target Range Register:

 Bit: |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
      | TR7 | TR6 | TR5 | TR4 | TR3 | TR2 | TR1 | TR0 |

TR7..TR0
  Target range value. Upon detection of a target (as
indicated by the status register), this register holds
a course estimate of the target range. This number's units are in
microseconds and it is accurate to within +/- one microsecond. A fixed
delay of six microseconds should be subtracted from each reading to account
for hardware delays.

*/

/* Register Defines */
#define LSRC_BASE 0x1000
#define LSRC_CONTROL 0x1000
#define LSRC_STATUS 0x1001
#define LSRC_BLANK  0x1002
#define LSRC_TARGET 0x1003
```

Appendix D

ABEL Code for LSRC Programmable Logic Devices

```
MODULE std_dcd
TITLE 'Source File: lsr_std_decoder.abl
JEDEC File: std_dcd.jed
Revision: 0.1
        Date: 3/20/95
        Designer: S. Boone'
"Build Commands: $ abel lsr_std_decoder.abl
"Unprogrammed Part Number: PAL22V10-15
"-------------------------------------------------------------
"CHANGE HISTORY
" Date          Inits.                Rev.    Comments
" -----------   ------                ----    --------
" 11-JUL-1994   spb                    0.0     First issue
" 20-MAR-1995   spb                    0.1     Changed address to 0x1ff.
"-------------------------------------------------------------
"
"DESCRIPTION
"
"  This device decodes the STD address for the LSRC.
" It receives a sixteen bit address from the STD bus along with
"the active low signal STD_IORQ*. When the address is correct
"and the STD_IORQ signal both are active, the output of this
"device will go active: MY_CARD*.
"-----------------------------------------------------

std_dcd        DEVICE 'p22v10' ;

DECLARATIONS
"
"INPUT PINS
"
          STD_IO_CLK            PIN       1;
          STD_IORQ_L            PIN       2;  "Active L
          STD_ADD4              PIN       3;
          STD_ADD5              PIN       4;
          STD_ADD6              PIN       5;
          STD_ADD7              PIN       6;
          STD_ADD8              PIN       7;
          STD_ADD9              PIN       8;
          STD_ADD10             PIN       9;
          STD_ADD11             PIN       10;
          STD_ADD12             PIN       11;
          STD_ADD13             PIN       13;
          STD_ADD14             PIN       14;
          STD_ADD15             PIN       15;
"
```

```
"OUTPUT PINS
"
        MY_CARD_L               PIN       20;


"Unused I/O
          UNUSED23          PIN   23;
          UNUSED22          PIN   22;
          UNUSED21          PIN   21;
          UNUSED19          PIN   19;
          UNUSED18          PIN   18;
          UNUSED17          PIN   17;
          UNUSED16          PIN     16;

"CONSTANTS
H,L,C,Z,X       =       1,0,.C.,.Z.,.X.;
STD_ADDRESS  = [STD_ADD15..STD_ADD4];

"Change the following parameter to change the decoded address
"for the card.
LSR_STD_ADDRESS = ^h1ff;
"-------------------------------------------------------------"

EQUATIONS
" The following assignment sets the STD address to the range of
"0x1ff0 through 0x1fff for a total of 16 possible single byte
"locations for this card.

MY_CARD_L = !((STD_ADDRESS==LSR_STD_ADDRESS) & !STD_IORQ_L);

"Assign unused I/O to H
          UNUSED23 = 1;
          UNUSED22 = 1;
          UNUSED21 = 1;
          UNUSED19 = 1;
          UNUSED18 = 1;
          UNUSED17 = 1;
          UNUSED16 = 1;

END std_dcd;
```

```
MODULE std_ctrl
TITLE 'Source File: lsr_std_control.abl
JEDEC File: std_ctrl.jed
Revision: 0.1
        Date: 3/20/95
        Designer: S. Boone'
"Build Commands:        $ abel4 lsr_std_control.abl
"Unprogrammed Part Number: MACH230a-20JC
"
"-------------------------------------------------------------------------
"CHANGE HISTORY
" Date          Inits.  Rev.    Comments
" -----------   ------  ----    --------
" 17-JUL-1993   spb     0.0     First issue
" 20-MAR-1995   spb     0.1     Added clock for STD write cycle 8 MHz
"-------------------------------------------------------------------------
"
"DESCRIPTION
"
"    This device controls the STD/DIP switch interface of the LSRC card.
"Its inputs are varied and are explained in the comments that follow.
 "-----------------------------------------------------------------------
         std_ctrl DEVICE 'MACH230a' ;

DECLARATIONS
"
"INPUT PINS
"
         CLOCK                  PIN       20;   "(A6)
         WR_CLOCK               PIN       62;

         TARGET_DET_L           PIN       41;   "(F13) Active Low
         BIT_START_L            PIN       65;   "(M7)  Active Low
         RDR_MLFNCTN_L          PIN       83;   "(H1)  Active Low
         STD_WR_L               PIN       3;    "(F1)  Active Low
         STD_RD_L               PIN       4;    "(F2)  Active Low
         MY_CARD_L              PIN       5;    "(E1)  Active Low
         STD_ADR0               PIN       6;    "(E2)
         STD_ADR1               PIN       7;    "(D1)
         STD_ADR2               PIN       8;    "(D2)
         STD_ADR3               PIN       9;    "(C1)
         STD_H                  PIN       10;   "(C2)
         BLANK0                 PIN       12;   "(A2)
         BLANK1                 PIN       13;   "(B2)
         BLANK2                 PIN       14;   "(A3)
         BLANK3                 PIN       15;   "(B3)
         BLANK4                 PIN       16;   "(A4)
         BLANK5                 PIN       17;   "(B4)
         BLANK6                 PIN       18;   "(A5)
         BLANK7                 PIN       19;   "(B5)
         CTRL0                  PIN       24;   "(A8)
         CTRL1                  PIN       25;   "(B8)
```

```
        CTRL2                PIN       26;  "(A9)
        CTRL3                PIN       27;  "(B9)
        NO_TRIG_L            PIN       28;  "(A10)
        TARGET_ADR0          PIN       29;  "(B10)
        TARGET_ADR1          PIN       30;  "(A11)
        TARGET_ADR2          PIN       31;  "(B11)
        TARGET_ADR3          PIN       33;  "(B13)
        TARGET_ADR4          PIN       34;  "(B12)
        TARGET_ADR5          PIN       35;  "(C13)
        TARGET_ADR6          PIN       36;  "(C12)
        TARGET_ADR7          PIN       37;  "(D13)
        STD_RESET_L          PIN       38;   "(D12) Active Low

        PUSH_RESET_NC        PIN       39;    "(E13)
        PUSH_RESET_NO        PIN       40;    "(E12)
        BIT_STATUS0          PIN       54;    "(N12)
        BIT_STATUS1          PIN       55;    "(M12)
        BIT_STATUS2          PIN       56;    "(N11)
"
"OUTPUT PINS
"
        STD_XCVR_OUT_L       PIN       45;    "(H13) Active Low
        CTRL_REG_RD_L        PIN       46;    "(H12) Active Low
        CTRL_REG_WR          PIN       47;    "(J13)
        BLANK_REG_RD_L       PIN       48;    "(J12) Active Low
        BLANK_REG_WR         PIN       49;    "(K13)
        STD_XCVR_EN_L        PIN       50;    "(K12)  Active Low
        BIT_FAIL_L           PIN       51;    "(L13) Active Low
        CARD_RESET_L         PIN       52;    "(L12) Active Low
        INT_DATA_OUT0        PIN       70;    "(N4)
        INT_DATA_OUT1        PIN       71;    "(M4)
        INT_DATA_OUT2        PIN       72;    "(N3)
        INT_DATA_OUT3        PIN       73;    "(M3)
        INT_DATA_OUT4        PIN       79;    "(K1)
        INT_DATA_OUT5        PIN       80;    "(K2)
        INT_DATA_OUT6        PIN       81;    "(J1)
        INT_DATA_OUT7        PIN       82;    "(J2)
"NODES
        INT_RESET1           NODE        ;
        INT_RESET2           NODE        ;
        AUTO_WRITE_CTRL      NODE        ;
        AUTO_WRITE_BLNK      NODE        ;
        AUTO_DATA_EN         NODE        ;
        AUTO_REG_SEL         NODE        ;
        WR_REGA              NODE        ;
        WR_REGB              NODE        ;
        DET_STD_WR           NODE        ;
        Q0                   NODE        ;
        Q1                   NODE        ;
        Q2                   NODE        ;
        DATA_OUT_OE          NODE        ;
```

```
"CONSTANTS
          H,L,C,Z,X        =        1,0,.C.,.Z.,.X.;
          STD_ADDRESS              =    [STD_ADR3..STD_ADR0];
          DATA_OUT                 =[INT_DATA_OUT7..INT_DATA_OUT0];
          STATUS_BITS = [1,NO_TRIG_L,STD_H,BIT_STATUS2..BIT_STATUS0,
                         RDR_MLFNCTN_L,TARGET_DET_L];
TSTV_STATUS = [BIT_STATUS2..BIT_STATUS0,RDR_MLFNCTN_L,TARGET_DET_L];
TARGET_BITS = [TARGET_ADR7..TARGET_ADR0];
CONTROL_BITS = [1,1,1,1,CTRL3..CTRL0];
TSTV_CTRL    = [CTRL3..CTRL0];
BLANK_BITS = [BLANK7..BLANK0];
NON_STD    = [Q2..Q0];
"
"  ------------------------------------------------------------------------
"
"ARCHITECTURE DEFINITIONS
"
          Q0                ISTYPE  'reg,buffer';
          Q1                ISTYPE  'reg,buffer';
          Q2                ISTYPE  'reg,buffer';
STD_XCVR_OUT_L  ISTYPE  'neg,buffer';
CTRL_REG_RD_L   ISTYPE  'neg,buffer';
BLANK_REG_RD_L  ISTYPE  'neg,buffer';
"  ------------------------------------------------------------------------
"
EQUATIONS
"Turn on the transceiver for driving the STD BUS only when the LSRC is
"selected by the STD and there is an STD READ cycle in progress:
STD_XCVR_OUT_L = !(!MY_CARD_L & !STD_RD_L);

"The feedback term in the following equation is to avoid bus contention
"on readback
STD_XCVR_EN_L = !(!STD_XCVR_OUT_L.fb # !STD_WR_L);

"Implement cross coupled NAND function to debounce PUSH BUTTON RESET:
INT_RESET1 = !(PUSH_RESET_NC & INT_RESET2);
INT_RESET2 = !(PUSH_RESET_NO & INT_RESET1);

"Issue a card reset if either the switch is pressed or
"a reset is issued by the STD
CARD_RESET_L = !(!INT_RESET1 # !STD_RESET_L);

"Detect the STD_WR signal with an 8 MHz clock and create a 125 ns
"strobe to the write signal on the latch
WR_REGA.clk = WR_CLOCK;
WR_REGB.clk = WR_CLOCK;
DET_STD_WR.clk = WR_CLOCK;

WR_REGA := STD_WR_L;
WR_REGB := WR_REGA;
DET_STD_WR := (!WR_REGA & WR_REGB);
```

```
"Issue register reads only in STD mode for the following readback registers
CTRL_REG_RD_L = !(STD_H & !MY_CARD_L & !STD_RD_L & (STD_ADDRESS==0));
BLANK_REG_RD_L = !(STD_H & !MY_CARD_L & !STD_RD_L & (STD_ADDRESS==2));

"Issue register writes differently depending on the STD switch. In non-STD
"mode, the write signal is generated by a state machine which is controls
the  "data to the register..
CTRL_REG_WR = (STD_H & !MY_CARD_L & !STD_WR_L & (STD_ADDRESS==0)) #
                  (!STD_H & AUTO_WRITE_CTRL.fb);
BLANK_REG_WR = (STD_H & !MY_CARD_L & !STD_WR_L & (STD_ADDRESS==2)) #
                  (!STD_H & AUTO_WRITE_BLNK.fb);

"Invert BIT_START_L for status. This is done such that if this device fails
"and is interrogated by BIT_START_L, it cannot respond with an active high
"output.
BIT_FAIL_L = !(BIT_START_L);
"Output data to the local (internal) bus to read the status or target
"registers in STD mode or to write the data to the control and blanking
"registers in non-std mode.
 DATA_OUT_OE = (STD_H & !MY_CARD_L & !STD_RD_L &
          ((STD_ADDRESS==1) # (STD_ADDRESS==3))) # (!STD_H &
            AUTO_DATA_EN.fb);

"Device does not allow multiple product terms on .oe equation.
DATA_OUT.oe = DATA_OUT_OE;
WHEN (STD_H & !STD_RD_L & (STD_ADDRESS==1)) THEN
      DATA_OUT = STATUS_BITS;
WHEN (STD_H & !STD_RD_L & (STD_ADDRESS==3)) THEN
      DATA_OUT = TARGET_BITS;
WHEN (!STD_H & !AUTO_REG_SEL.fb) THEN DATA_OUT = CONTROL_BITS;
WHEN (!STD_H & AUTO_REG_SEL.fb) THEN
      DATA_OUT = BLANK_BITS;

"Define clocks for state diagram to be executed after a reset in NON STD
mode "to read the values of the DIP switches to load the control and
blanking "registers
Q0.clk  =  CLOCK;
Q1.clk  =  CLOCK;
Q2.clk  =  CLOCK;
AUTO_REG_SEL.clk = CLOCK;
AUTO_DATA_EN.clk = CLOCK;
AUTO_WRITE_CTRL.clk = CLOCK;
AUTO_WRITE_BLNK.clk = CLOCK;

STATE_DIAGRAM NON_STD
STATE 0:
AUTO_REG_SEL := 1;
AUTO_DATA_EN := 0;
AUTO_WRITE_CTRL := 0;
AUTO_WRITE_BLNK := 0;
if ((CARD_RESET_L==0) & (STD_H==0)) then 1 else 0;
```

```
STATE 1:
AUTO_REG_SEL := 1;
AUTO_DATA_EN := 0;
AUTO_WRITE_CTRL := 0;
AUTO_WRITE_BLNK := 0;
goto 2;

" Output enable the data on the internal bus and turn on the blanking data
"bits at the output and make the blanking register transparent:
STATE 2:
AUTO_REG_SEL := 1;
AUTO_DATA_EN := 1;
AUTO_WRITE_CTRL := 0;
AUTO_WRITE_BLNK := 1;
goto 3;

"Latch the blanking register data in:
STATE 3:
AUTO_REG_SEL := 1;
AUTO_DATA_EN := 1;
AUTO_WRITE_CTRL := 0;
AUTO_WRITE_BLNK := 0;
goto 4;

"Now switch the output to the control input:
STATE 4:
AUTO_REG_SEL := 0;
AUTO_DATA_EN := 1;
AUTO_WRITE_CTRL := 1;
AUTO_WRITE_BLNK := 0;
goto 5;

"Latch the Control data:
STATE 5:
AUTO_REG_SEL := 0;
AUTO_DATA_EN := 1;
AUTO_WRITE_CTRL := 0;
AUTO_WRITE_BLNK := 0;
goto 6;

"Now turn off the output enable and stay here until reset releases:
 STATE 6:
AUTO_REG_SEL := 1;
AUTO_DATA_EN := 0;
AUTO_WRITE_CTRL := 0;
AUTO_WRITE_BLNK := 0;
if (CARD_RESET_L==1) then 0 else 6;

STATE 7: goto 0;

END std_ctrl;
```

```
MODULE lsr_ckgn
TITLE 'Source File: lsr_clock_divider.abl
JEDEC File: lsr_ckgn.jed
Revision: 0.1
        Date: 3/20/95
        Designer: S. Boone'
"Build Commands: $ abel lsr_clock_divider.abl
"Unprogrammed Part Number: PAL22V10-15

"-------------------------------------------------------------------------
"CHANGE HISTORY
" Date          Inits.                  Rev.    Comments
" 10-JUL-1994   spb                     0.0     First issue
" 20-MAR-1995   spb                 ....0.1     Added second set of clocks

"-------------------------------------------------------------------------
"
"DESCRIPTION
"
"   This device controls the clocks for the LSRC. It receives a clock input
"on pin 1 and divides it by two, four, eight and sixteen. The clocks reset
"only on power up. The RESET input is provided for later expansion. Two
"sets of clocks are provided.
"-------------------------------------------------------------------------

lsr_ckgn        DEVICE 'p22v10' ;

DECLARATIONS

"
"INPUT PINS
"
         CLOCK_16M          PIN     1;  "
         RESET              PIN     2;  "Active L - NOT USED
         BIT_START_L        PIN    13;
"
"OUTPUT PINS
"
         CNTA0                      PIN        23; "8 MHZ CLOCK
         CNTA1                      PIN        22; "4 MHZ CLOCK
         CNTA2                      PIN        21; "2 MHZ CLOCK
         CNTA3                      PIN        20; "1 MHZ CLOCK
         CNTA4                      PIN        19; "500 Khz CLOCK
         CNTB0                      PIN        18;
         CNTB1                      PIN        17;
         CNTB2                      PIN        16;

         CNTB3                      PIN        15;

         BIT_FAIL_L                 PIN        14;

"CONSTANTS
```

```
          H,L,C,Z,X       =           1,0,.C.,.Z.,.X.;
           COUNTA    = [CNTA4..CNTA0];
           COUNTB    = [CNTB3..CNTB0];

"  -------------------------------------------------------------------
"
EQUATIONS

"Define a 4 bit 16M down counter. The counter is reset on
"power up so that all of the clocks start low.

COUNTA.clk = CLOCK_16M;
COUNTA := (COUNTA.fb - 1);

COUNTB.clk = CLOCK_16M;
COUNTB := (COUNTB.fb - 1);

"Invert BIT_START_L for status. This is done such that if this device fails
"and is interrogated by BIT_START_L, it cannot respond with an active high
"output.
BIT_FAIL_L = !(BIT_START_L);

END lsr_ckgn;
```

```
MODULE stg_ctrl
TITLE 'Source File: lsr_stagger_control.abl
JEDEC File: stg_ctrl.jed
        Revision: 0.0
        Date: 7/11/94
        Designer: S. Boone'
"Build Commands: $ abel lsr_stagger_control.abl
"Unprogrammed Part Number: PAL22V10-15
"-------------------------------------------------------------------------
"CHANGE HISTORY
" Date          Inits.                Rev.    Comments
" 11-JUL-1994   spb                    0.0     First issue
"-------------------------------------------------------------------------
"
"DESCRIPTION
"
" This device controls the two stage stagger circuit for the LSRC. It
"receives as input a 500 Khz clock, the original 50 percent duty cycle PRF
"signal from the Raytheon radar and two control bits to choose the delay
"time for the staggered or dithered pulse. The input PRF is double buffered
"to avoid metastability problems associated with registering an
"asynchronous signal. This process alone yields a degree of pseudo-
"randomness to the final output PRF. Every other pulse received is then
"delayed zero to six additional microseconds according to the control bits,
"ie, 0 = zero microseconds, 1 = 2 microsecond, 2 = 4 microseconds and 3 = 6
"microseconds. An additional delayed signal is provided for future
"expansion if desired. The device also responds with a BIT_FAIL_L signal in
"response to a BIT_START_L input as a Built in Test function for the LSRC.
"-------------------------------------------------------------------------

stg_ctrl        DEVICE 'p22v10' ;

DECLARATIONS
"
"INPUT PINS
"
          CLK_IN            PIN     1;
          RESET             PIN     2;  "Active L
          STAGGER_CTRL0     PIN     3;
          STAGGER_CTRL1     PIN     4;
          ORIGINAL_PRF      PIN     5;
          BIT_START_L       PIN    13;
"
"OUTPUT PINS
"
          STAGGERED_PRF             PIN       23;
          PRF_DLD_0                 PIN       22;
          PRF_DLD_1                 PIN       21;
          PRF_DLD_2                 PIN       20;
          PRF_DLD_3                 PIN       19;
          PRF_DLD_4                 PIN       18;
          PRF_DLD_5                 PIN       17;
```

```
          EDGE_DET_PRF              PIN       16;
          EDGE_DET_CNT              PIN       15;
          BIT_FAIL_L                PIN        14;

"CONSTANTS
          H,L,C,Z,X       =        1,0,.C.,.Z.,.X.;
          STG_CTRL = [STAGGER_CTRL1..STAGGER_CTRL0];
"   -------------------------------------------------------------------
"

EQUATIONS
PRF_DLD_0.clk = CLK_IN;
PRF_DLD_1.clk = CLK_IN;
PRF_DLD_2.clk = CLK_IN;
PRF_DLD_3.clk = CLK_IN;
PRF_DLD_4.clk = CLK_IN;
PRF_DLD_5.clk = CLK_IN;
EDGE_DET_PRF.clk = CLK_IN;
EDGE_DET_CNT.clk = CLK_IN;
STAGGERED_PRF.clk = CLK_IN;
PRF_DLD_0 := RESET & ORIGINAL_PRF;
PRF_DLD_1 := RESET & PRF_DLD_0.fb;
PRF_DLD_2 := RESET & PRF_DLD_1.fb;
PRF_DLD_3 := RESET & PRF_DLD_2.fb;
PRF_DLD_4 := RESET & PRF_DLD_3.fb;
PRF_DLD_5 := RESET & PRF_DLD_4.fb;

"Edge detect the falling edge of the most delayed signals
"allow time for the signal to switch over to the second stage of the
"stagger algorithm:
EDGE_DET_PRF := (RESET & !PRF_DLD_3.fb & PRF_DLD_4.fb);

EDGE_DET_CNT := (RESET & (!EDGE_DET_CNT.fb & EDGE_DET_PRF.fb) #
                (EDGE_DET_CNT.fb & !EDGE_DET_PRF.fb));

STAGGERED_PRF := (RESET & ((!EDGE_DET_CNT.fb & PRF_DLD_1.fb) #
                 (EDGE_DET_CNT.fb & (STG_CTRL==0) & PRF_DLD_1.fb) #
                 (EDGE_DET_CNT.fb & (STG_CTRL==1) & PRF_DLD_2.fb) #
                 (EDGE_DET_CNT.fb & (STG_CTRL==2) & PRF_DLD_3.fb) #
                 (EDGE_DET_CNT.fb & (STG_CTRL==3) & PRF_DLD_4.fb)));


"Invert BIT_START_L for status. This is done such that if this device fails
"and is interrogated "by BIT_START_L, it cannot respond with an active high
"output.
BIT_FAIL_L = !(BIT_START_L);

END stg_ctrl;
```

```
MODULE thr_ctrl
TITLE 'Source File: lsr_threshold_control.abl
        JEDEC File: thr_ctrl.jed
        Revision: 0.0
        Date: 7/17/94
        Designer: S. Boone'
"Build Commands:$ abel4 lsr_threshold_control.abl
"Unprogrammed Part Number: MACH230a-20JC
"-------------------------------------------------------------------------
"CHANGE HISTORY
" Date          Inits.  Rev.    Comments
" 17-JUL-1993   spb     0.0     First issue
"-------------------------------------------------------------------------
"
"DESCRIPTION
"
"    This device controls the threshold algorithm and BIT and display
"functions for the LSRC card. Further comments are given with the output
"equations where appropriate.
"-------------------------------------------------------------------------

        thr_ctrl DEVICE 'MACH230a' ;

DECLARATIONS
"
"INPUT PINS
"
        CLOCK_4M            PIN     20;  "(A6)
        RESET               PIN     41;  "(F13) Active Low
        TRIG_DET_L          PIN     83;  "(H1)  Active Low
        THR_ALG_ON          PIN     3;   "(F1)
        STD_CTRL_BTFL_L     PIN     13;  "(B2)
        CLK_DIV_BTFL_L      PIN     14;  "(A3)
        STG_CTRL_BTFL_L     PIN     15;  "(B3)
        RNG_BIN_BTFL_L      PIN     16;  "(A4)
        TARGET_DET_L        PIN     24;  "(A8)
        RADAR_MLFNCTN_L     PIN     25;  "(B8)
        RAM_FAIL_L          PIN     26;  "(A9)
"
"OUTPUT PINS
"
        BIT_STATUS0         PIN     38;  "(D12)
        BIT_STATUS1         PIN     39;  "(E13)
        BIT_STATUS2         PIN     40;  "(E12)

                                         "Bit weights:
        THRESH_COUNT0       PIN     45;  "(H13)  250 ns
        THRESH_COUNT1       PIN     46;  "(H12)  500 ns
        THRESH_COUNT2       PIN     47;  "(J13)    1 us
        THRESH_COUNT3       PIN     48;  "(J12)    2 us
        THRESH_COUNT4       PIN     49;  "(K13)    4 us
        THRESH_COUNT5       PIN     50;  "(K12)    8 us
```

```
        THRESH_COUNT6          PIN       51;   "(L13)  16 us
        THRESH_COUNT7          PIN       52;   "(L12)  32 us
        THRESH_COUNT8          PIN       54;   "(N12)  64 us
        THRESH_COUNT9          PIN       55;   "(M12) 128 us
        THRESH_COUNT10         PIN       56;   "(N11) 256 us
        THRESH_COUNT11         PIN       57;   "(M11) 512 us
        THRESH_COUNT12         PIN       58;   "(N10) 1.024 ms
        THRESH_COUNT13         PIN       59;   "(M10) 2.048 ms

        THRESH_CS_L            PIN       66;     "(N6) Active Low
        STD_CTRL_BTST_L        PIN       67;     "(M6) Active Low
        CLK_DIV_BTST_L PIN               68;     "(N5) Active Low
        STG_CTRL_BTST_L        PIN       69;     "(M5) Active Low
        RNG_BIN_BTST_L         PIN       70;     "(N4) Active Low
        BIT_SHUTDOWN_L         PIN       71;     "(M4) Active Low

        SEGP                   PIN       75;     "(M1)
        SEGC                   PIN       76;     "(M2)
        SEGB                   PIN       77;     "(L1)
        SEGA                   PIN       78;     "(L2)
        SEGG                   PIN       79;     "(K1)
        SEGF                   PIN       80;     "(K2)
        SEGE                   PIN       81;     "(J1)
        SEGD                   PIN       82;     "(J2)
"NODES
        TRIG_REG1              NODE        ;
        TRIG_REG2              NODE        ;
        TRIG_EDGE_DET          NODE        ;
        BIT5MAX                NODE        ;
        BIT11MAX               NODE        ;
        FIRST_TRIG             NODE        ;
        Q0                     NODE        ;
        Q1                     NODE        ;
        Q2                     NODE        ;
        Q3                     NODE        ;
        R0                     NODE        ;
        R1                     NODE        ;
        R2                     NODE        ;
        R3                     NODE        ;

"CONSTANTS
        H,L,C,Z,X          =          1,0,.C.,.Z.,.X.;
        ZERO = 0;

COUNT_5     = [THRESH_COUNT5..THRESH_COUNT0].fb;
COUNT_11    = [THRESH_COUNT11..THRESH_COUNT6].fb;
LSB_COUNTS = [THRESH_COUNT2..THRESH_COUNT0].fb;
THRESH_COUNTER = [THRESH_COUNT13..THRESH_COUNT0];
BIT         = [Q3..Q0];
BIT_STATUS = [BIT_STATUS2..BIT_STATUS0];
DISPLAY     = [R3..R0];
```

```
" Zero = segment on. One = segment off
" Note that decimal point on indicates a failed state if the display is
" non-zero and indicates a holding state if the display is zero. It is
"handled directly in the DISPLAY state diagram
DISPLAY_VAL = [SEGA,SEGB,SEGC,SEGD,SEGE,SEGF,SEGG];
AOK_DIS    =  [          0,      0,    0,      0,    0,      0,     1];"OUTPUT 0
TARGET_DIS=   [          1,      0,    0,      1,    1,      1,     1];"OUTPUT 1
MLFCTN_DIS=   [          0,      0,    1,      0,    0,      1,     0];"OUTPUT 2
RAMFL_DIS =   [          0,      0,    0,      0,    1,      1,     0];"OUTPUT 3
BTFLA_DIS=    [          0,      0,    0,      1,    0,      0,     0];"OUTPUT A
BTFLB_DIS=    [          1,      1,    0,      0,    0,      0,     0];"OUTPUT b
BTFLC_DIS=    [          1,      1,    1,      0,    0,      1,     0];"OUTPUT c
BTFLD_DIS=    [          1,      0,    0,      0,    0,      1,     0];"OUTPUT d
BTBAD_DIS=    [           0,      0,    0,     0,    0,       0   ,0];"OUTPUT 8
"  ------------------------------------------------------------------------
"
"ARCHITECTURE DEFINITIONS
"
          THRESH_COUNTER  ISTYPE  'reg_t,buffer';
          Q0                 ISTYPE  'reg,buffer';
          Q1                 ISTYPE  'reg,buffer';
          Q2                 ISTYPE  'reg,buffer';
          Q3                 ISTYPE  'reg,buffer';

          STD_CTRL_BTST_L ISTYPE  'reg,neg,buffer';
          CLK_DIV_BTST_L  ISTYPE  'reg,neg,buffer';
          STG_CTRL_BTST_L ISTYPE  'reg,neg,buffer';
          RNG_BIN_BTST_L  ISTYPE  'reg,neg,buffer';

          R0                 ISTYPE  'reg,buffer';
          R1                 ISTYPE  'reg,buffer';
          R2                 ISTYPE  'reg,buffer';
          R3                 ISTYPE  'reg,buffer';
          THRESH_CS_L        ISTYPE  'neg,buffer';
"-------------------------------------------------------------------------
"
EQUATIONS
"Edge detect the trigger signal from the radar:
TRIG_REG1.clk = CLOCK_4M;
TRIG_REG2.clk = CLOCK_4M;
TRIG_EDGE_DET.clk = CLOCK_4M;
TRIG_REG1 := !(RESET & !TRIG_DET_L);
TRIG_REG2 := !(RESET & !TRIG_REG1);
TRIG_EDGE_DET := RESET & !TRIG_REG1 & TRIG_REG2;

"Use the following signal to hold the counter at zero after a reset until "
the first trigger is encountered.
          FIRST_TRIG.clk = CLOCK_4M;
          FIRST_TRIG := RESET & (TRIG_EDGE_DET # FIRST_TRIG);
```

```
"Set up the 12 bit counter. The counter free runs between triggers and
"resets to zero on each trigger or on reset. The counter is held at zero
"until the first trigger is detected and will stay at zero if the threshold
"algorithm is off.
          BIT5MAX =  (COUNT_5.fb==63);
          BIT11MAX = (COUNT_11.fb==63);
          THRESH_COUNTER.clk = CLOCK_4M;

WHEN
(!RESET # TRIG_EDGE_DET # !THR_ALG_ON)
THEN
THRESH_COUNTER.t = (THRESH_COUNTER.fb $ ZERO);


THRESH_COUNT13.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                    BIT5MAX  & BIT11MAX & ([THRESH_COUNT12].fb == 1));

THRESH_COUNT12.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                    BIT5MAX & BIT11MAX);

THRESH_COUNT11.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                    BIT5MAX & ([THRESH_COUNT10..THRESH_COUNT6].fb == 31));

THRESH_COUNT10.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                    BIT5MAX & ([THRESH_COUNT9..THRESH_COUNT6].fb == 15));

THRESH_COUNT9.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                    BIT5MAX & ([THRESH_COUNT8..THRESH_COUNT6].fb == 7));

THRESH_COUNT8.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                    BIT5MAX & ([THRESH_COUNT7..THRESH_COUNT6].fb == 3));

THRESH_COUNT7.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                       BIT5MAX & ([THRESH_COUNT6].fb == 1));

THRESH_COUNT6.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                       BIT5MAX);

THRESH_COUNT5.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                       ([THRESH_COUNT4..THRESH_COUNT0].fb == 31));

THRESH_COUNT4.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                       ([THRESH_COUNT3..THRESH_COUNT0].fb == 15));

THRESH_COUNT3.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                       ([THRESH_COUNT2..THRESH_COUNT0].fb == 7));

THRESH_COUNT2.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                       ([THRESH_COUNT1..THRESH_COUNT0].fb == 3));

THRESH_COUNT1.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON &
                       ([THRESH_COUNT0].fb == 1));
```

```
THRESH_COUNT0.t = (RESET & !TRIG_EDGE_DET & FIRST_TRIG & THR_ALG_ON);

"Output the D/A CS signal for ~750 ns for each 2 us interval. The Threshold
"PROM is addressed using the upper 9 bits of the 12 bit counter. Use the
"RESET signal to create a CS signal when the Threshold algorithm is off.
"This will latch the data at PROM address zero into the D/A.The high order
"address bit qualification holds the data in the D/A after 512 us since no
"CS signals will be generated after this until the next trigger.
THRESH_CS_L.clk = CLOCK_4M;
THRESH_CS_L := !(!RESET # (!THRESH_COUNT11 &
                      (LSB_COUNTS==3)#(LSB_COUNTS==4)#(LSB_COUNTS==5)));

"Set up clocks and the state diagram for the BIT. BIT will individually
"exercise inputs of four other devices on the card and wait for a response.
"If an error is encountered, the error condition is latched and a shutdown
"action is initiated. A code indicating the suspect component is issued to
"the display and to the status register.
Q0.clk = CLOCK_4M;
Q1.clk = CLOCK_4M;
Q2.clk = CLOCK_4M;
Q3.clk = CLOCK_4M;

STD_CTRL_BTST_L.clk = CLOCK_4M;
CLK_DIV_BTST_L.clk = CLOCK_4M;
STG_CTRL_BTST_L.clk = CLOCK_4M;
RNG_BIN_BTST_L.clk = CLOCK_4M;
BIT_SHUTDOWN_L = !(RESET & (BIT_STATUS!=0));

STATE_DIAGRAM BIT
STATE 0:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = 0;
 if (RESET==0) then 0 else 1;

STATE 1:
 STD_CTRL_BTST_L := 0;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = 0;
if (RESET==0) then 0 else
   if (RAM_FAIL_L==0) then 14 else 2;

STATE 2:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = 0;
```

```
if (STD_CTRL_BTFL_L==0) then 10 else if (RESET==0) then 0 else 3;
STATE 3:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 0;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = 0;
if (STD_CTRL_BTFL_L==1) then 10 else if (RESET==0) then 0 else 4;

STATE 4:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = 0;
if (CLK_DIV_BTFL_L==0) then 11 else if (RESET==0) then 0 else 5;

STATE 5:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 0;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = 0;
if (CLK_DIV_BTFL_L==1) then 11 else if (RESET==0) then 0 else 6;

STATE 6:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = 0;
if (STG_CTRL_BTFL_L==0) then 12 else if (RESET==0) then 0 else 7;

STATE 7:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 0;
 BIT_STATUS = 0;
if (STG_CTRL_BTFL_L==1) then 12 else if (RESET==0) then 0 else 8;

STATE 8:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = 0;
if (RNG_BIN_BTFL_L==0) then 13 else
    if (RESET==0) then 0 else 9;

STATE 9:
 STD_CTRL_BTST_L := 1;
```

```
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = 0;
if (RNG_BIN_BTFL_L==1) then 13 else
    if (RESET==0) then 0 else 1;

"If failure is detected, stay in failed state until reset.
STATE 10:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = [0,0,1];
if (RESET==0) then 0 else 10;

STATE 11:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = [0,1,0];
if (RESET==0) then 0 else 11;

STATE 12:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = [0,1,1];
if (RESET==0) then 0 else 12;

STATE 13:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = [1,0,0];
if (RESET==0) then 0 else 13;

STATE 14:
 STD_CTRL_BTST_L := 1;
 CLK_DIV_BTST_L  := 1;
 STG_CTRL_BTST_L := 1;
 RNG_BIN_BTST_L  := 1;
 BIT_STATUS = [1,0,1];
if (RESET==0) then 0 else 14;

STATE 15: goto 0;
```

```
EQUATIONS
"Set up the state diagram for the display. Once anything is displayed other
"than a good status, that state is held until a reset is issued.
R0.clk = CLOCK_4M;
R1.clk = CLOCK_4M;
R2.clk = CLOCK_4M;
R3.clk = CLOCK_4M;

STATE_DIAGRAM DISPLAY

STATE 0:
DISPLAY_VAL = AOK_DIS;
SEGP = FIRST_TRIG;
if (RESET==0) then 0 else 1;

STATE 1:
DISPLAY_VAL = AOK_DIS;
SEGP = FIRST_TRIG;
if (TARGET_DET_L==0) then 8 else
 if (RESET==0) then 0 else 2;

STATE 2:
DISPLAY_VAL = AOK_DIS;
SEGP = FIRST_TRIG;
if (RADAR_MLFNCTN_L==0) then 9 else
 if (RESET==0) then 0 else 3;

STATE 3:
DISPLAY_VAL = AOK_DIS;
SEGP = FIRST_TRIG;
if (BIT_STATUS==1) then 10 else
 if (RESET==0) then 0 else 4;

STATE 4:
DISPLAY_VAL = AOK_DIS;
SEGP = FIRST_TRIG;
if (BIT_STATUS==2) then 11 else
 if (RESET==0) then 0 else 5;

STATE 5:
DISPLAY_VAL = AOK_DIS;
SEGP = FIRST_TRIG;
if (BIT_STATUS==3) then 12 else
 if (RESET==0) then 0 else 6;

STATE 6:
DISPLAY_VAL = AOK_DIS;
SEGP = FIRST_TRIG;
if (BIT_STATUS==4) then 13 else
 if (RESET==0) then 0 else 7;
```

```
STATE 7:
DISPLAY_VAL = AOK_DIS;
SEGP = FIRST_TRIG;
if (BIT_STATUS==5) then 14 else
   if (BIT_STATUS>5) then 15 else
if (RESET==0) then 0 else 1;

STATE 8:
DISPLAY_VAL = TARGET_DIS;
SEGP = 0;
if (RESET==0) then 0 else 8;

STATE 9:
DISPLAY_VAL = MLFCTN_DIS;
SEGP = 0;
if (RESET==0) then 0 else 9;

STATE 10:
DISPLAY_VAL = BTFLA_DIS;
SEGP = 0;
if (RESET==0) then 0 else 10;

STATE 11:
DISPLAY_VAL = BTFLB_DIS;
SEGP = 0;
if (RESET==0) then 0 else 11;

STATE 12:
DISPLAY_VAL = BTFLC_DIS;
SEGP = 0;
if (RESET==0) then 0 else 12;

STATE 13:
DISPLAY_VAL = BTFLD_DIS;
SEGP = 0;
if (RESET==0) then 0 else 13;

STATE 14:
DISPLAY_VAL = RAMFL_DIS;
SEGP = 0;
if (RESET==0) then 0 else 14;

STATE 15:
DISPLAY_VAL = BTBAD_DIS;
SEGP = 0;
if (RESET==0) then 0 else 15;


END thr_ctrl;
```

```
MODULE rng_ctrl
TITLE 'Source File: lsr_range_bin_control.abl
        JEDEC File: rng_ctrl.jed
        Revision: 0.0
        Date: 7/19/94
        Designer: S. Boone'
"Build Commands:        $ abel4 lsr_range_bin_control.abl
"Unprogrammed Part Number: MACH230a-20JC
" "-----------------------------------------------------------------
"CHANGE HISTORY
" Date          Inits.  Rev.    Comments
" 19-JUL-1993    spb     0.0     First issue
"-----------------------------------------------------------------
"
"DESCRIPTION
"
"    This device controls the blanking and range bin algorithms for the
" LSRC card. Further comments are given with the output equations where
" appropriate.
"-------------------------------------------------------

        rng_ctrl DEVICE 'MACH230a' ;

DECLARATIONS
"
"INPUT PINS
"
        CLOCK_8M            PIN      20;  "(A6)
        RANGE_3RD_PASS      PIN      23;  "(B7)
        RESET               PIN      41;  "(F13) Active Low
        BIT_START_L         PIN      65;  "(M7)  Active Low
        TRIG_DET_L          PIN      83;  "(H1)  Active Low

        TARGET_RET_L        PIN      3;    "(F1)
        BIT_SHUTDOWN_L      PIN      4;    "(F2)

        BLANK0              PIN     12;    "(A2)
        BLANK1              PIN     13;    "(B2)
        BLANK2              PIN     14;    "(A3)
        BLANK3              PIN     15;    "(B3)
        BLANK4              PIN     16;    "(A4)
        BLANK5              PIN     17;    "(B4)
        BLANK6              PIN     18;    "(A5)
        BLANK7              PIN     19;    "(B5)

        RAM_IN0             PIN     37;    "(D13)
        RAM_IN1             PIN     38;    "(D12)
        RAM_IN2             PIN     39;    "(E13)
        RAM_IN3             PIN     40;    "(E12)
```

```
"
"OUTPUT PINS
"
          TARGET_ADDR0          PIN       24;   "(A8)
          TARGET_ADDR1          PIN       25;   "(B8)
          TARGET_ADDR2          PIN       26;   "(A9)
          TARGET_ADDR3          PIN       27;   "(B9)
          TARGET_ADDR4          PIN       28;   "(A10)
          TARGET_ADDR5          PIN       29;   "(B10
          TARGET_ADDR6          PIN       30;   "(A11)
          TARGET_ADDR7          PIN       31;   "(B11)

          RAM_FAIL_L            PIN       36;    "(C12)

"Bit weight for the following counter is given for reference:
          RNGCOUNT0        PIN       45;   "(H13) 125       ns
          RNGCOUNT1        PIN       46;   "(H12) 250       ns
          RNGCOUNT2        PIN       47;   "(J13) 500       ns
          RNGCOUNT3        PIN       48;   "(J12) 1         us
          RNGCOUNT4        PIN       49;   "(K13) 2         us
          RNGCOUNT5        PIN       50;   "(K12) 4         us
          RNGCOUNT6        PIN       51;   "(L13) 8         us
          RNGCOUNT7        PIN       52;   "(L12) 16        us
          RNGCOUNT8        PIN       54;   "(N12) 32        us
          RNGCOUNT9        PIN       55;   "(M12) 64        us
          RNGCOUNT10       PIN       56;   "(N11) 128       us
          RNGCOUNT11       PIN       57;   "(M11) 256       us
          RNGCOUNT12       PIN       58;   "(N10) 512       us
          RNGCOUNT13       PIN       59;   "(M10) 1.024     ms
          RNGCOUNT14       PIN       60;   "(N9)  2.048     ms

          RAM_WR_L              PIN       61; "(M9) Active Low
          RAM_CS_L              PIN       66; "(N6) Active Low
          LASER_EN              PIN       67; "(M6) Active Low
          RADAR_MLFNCTN_L       PIN       68; "(N5) Active Low
          TARGET_L              PIN       69; "(M5) Active Low

          BIT_FAIL_L            PIN       78; "(L2)

          RAM_OUT0              PIN       79; "(K1)
          RAM_OUT1              PIN       80; "(K2)
          RAM_OUT2              PIN       81; "(J1)
          RAM_OUT3              PIN       82; "(J2)

          FIRST_TRIG            PIN       35; "(C13)


"NODES

          TRIG_REG1              NODE        ;
          TRIG_REG2              NODE        ;
          TRIG_EDGE_DET          NODE        ;
```

```
        TARG_REG1               NODE        ;
        TARG_REG2               NODE        ;
        TARG_EDGE_DET           NODE        ;

        MAXTO4                  NODE        ;
        MAXTO9                  NODE        ;

        BLANK_COMP              NODE        ;
        TARG_DET_EN             NODE        ;

        SEARCH_A                NODE        ;
        SEARCH_B                NODE        ;

        TARG_FOUND_A            NODE        ;
        TARG_FOUND_B            NODE        ;

        STATE_5                 NODE        ;

        RAM_READ_L              NODE        ;

        RAM_READ_DATA0          NODE        ;
        RAM_READ_DATA1          NODE        ;
        RAM_READ_DATA2          NODE        ;
        RAM_READ_DATA3          NODE        ;

        REG_TARGET_L            NODE        ;
        Q0                      NODE        ;
        Q1                      NODE        ;
        Q2                      NODE        ;

"CONSTANTS

        H,L,C,Z,X           =         1,0,.C.,.Z.,.X.;
        ZERO = 0;
COUNT_4     = [RNGCOUNT4..RNGCOUNT0].fb;
COUNT_9     = [RNGCOUNT9..RNGCOUNT5].fb;
LSB_COUNTS = [RNGCOUNT2..RNGCOUNT0].fb;
RNGCOUNTER = [RNGCOUNT14..RNGCOUNT0];
RNG_CNT_LBYTE = [RNGCOUNT7..RNGCOUNT0].fb;
LOWER4 = [RNGCOUNT3..RNGCOUNT0];
BLANK = [BLANK7..BLANK0];
RAM_IN = [RAM_IN3..RAM_IN0];
RAM_READ_DATA = [RAM_READ_DATA3..RAM_READ_DATA0];
RAM_OUT= [RAM_OUT3..RAM_OUT0];
TARGET_ADDR = [TARGET_ADDR7..TARGET_ADDR0];
TARGET_DET_ADDR = [RNGCOUNT10..RNGCOUNT3];
RAM_BIT = [Q2..Q0];
```

```
"
"ARCHITECTURE DEFINITIONS
"
          RNGCOUNTER          ISTYPE  'reg_t,buffer';
          Q0                  ISTYPE  'reg,buffer';
          Q1                  ISTYPE  'reg,buffer';
          Q2                  ISTYPE  'reg,buffer';
          RAM_WR_L            ISTYPE  'reg,neg,buffer';
          RAM_CS_L            ISTYPE  'reg,neg,buffer';
          LASER_EN            ISTYPE  'reg,neg,buffer';
          RADAR_MLFNCTN_L     ISTYPE  'reg,neg,buffer';
          BIT_FAIL_L          ISTYPE  'neg,buffer';
          TARGET_L            ISTYPE  'reg,neg,buffer';
"         RAM_OUT0            ISTYPE  'reg,neg,buffer';
"         RAM_OUT1            ISTYPE  'reg,neg,buffer';
"  ------------------------------------------------------------------
"
EQUATIONS

"Edge detect the trigger signal from the radar:
TRIG_REG1.clk = CLOCK_8M;
TRIG_REG2.clk = CLOCK_8M;
TRIG_EDGE_DET.clk = CLOCK_8M;
TRIG_REG1 := !(RESET & !TRIG_DET_L);
TRIG_REG2 := !(RESET & !TRIG_REG1);
TRIG_EDGE_DET := RESET & !TRIG_REG1 & TRIG_REG2;
"Edge detect the target signal from the radar:
TARG_REG1.clk = CLOCK_8M;
TARG_REG2.clk = CLOCK_8M; TARG_EDGE_DET.clk = CLOCK_8M;
TARG_REG1 := !(RESET & !TARGET_RET_L);
TARG_REG2 := !(RESET & !TARG_REG1);
TARG_EDGE_DET := RESET & !TARG_REG1 & TARG_REG2;

"Use the following signal to hold the counter at zero after a reset until
"the first trigger is encountered.
FIRST_TRIG.clk = CLOCK_8M;
FIRST_TRIG := RESET & ((TRIG_EDGE_DET) # FIRST_TRIG);

"Set up the 15 bit counter. The counter free runs between triggers and
"resets to zero on each trigger or on reset. The counter is held at zero
"until the first trigger is detected.
          MAXTO4 = (COUNT_4.fb==31);
          MAXTO9 = (COUNT_9.fb==31);

          RNGCOUNTER.clk = CLOCK_8M;

          WHEN (!RESET # (TRIG_EDGE_DET)) THEN
                    RNGCOUNTER.t = (RNGCOUNTER.fb $ ZERO);

RNGCOUNT14.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG &
                MAXTO4 & MAXTO9 & ([RNGCOUNT13..RNGCOUNT10].fb == 15));
```

```
RNGCOUNT13.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG &
                MAXTO4 & MAXTO9 & ([RNGCOUNT12..RNGCOUNT10].fb == 7));


RNGCOUNT12.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG &
                MAXTO4 & MAXTO9 & ([RNGCOUNT11..RNGCOUNT10].fb == 3));

RNGCOUNT11.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG &
                MAXTO4 & MAXTO9 & ([RNGCOUNT10].fb == 1));

RNGCOUNT10.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG &
                MAXTO4 & MAXTO9);

RNGCOUNT9.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG & MAXTO4 &
                       ([RNGCOUNT8..RNGCOUNT5].fb == 15));

RNGCOUNT8.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG & MAXTO4 &
                       ([RNGCOUNT7..RNGCOUNT5].fb == 7));

RNGCOUNT7.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG & MAXTO4 &
                       ([RNGCOUNT6..RNGCOUNT5].fb == 3));

RNGCOUNT6.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG & MAXTO4 &
                       ([RNGCOUNT5].fb==1));

RNGCOUNT5.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG & MAXTO4);

RNGCOUNT4.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG &
                       ([RNGCOUNT3..RNGCOUNT0].fb == 15));

RNGCOUNT3.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG &
                       ([RNGCOUNT2..RNGCOUNT0].fb == 7));

RNGCOUNT2.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG &
                        ([RNGCOUNT1..RNGCOUNT0].fb == 3));

RNGCOUNT1.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG &
                        ([RNGCOUNT0].fb == 1));

RNGCOUNT0.t = (RESET & !(TRIG_EDGE_DET) & FIRST_TRIG);

"Implement a blanking algorithm based on the blanking value input.
"Targets will be ignored until the end of the blanking interval for each
"cycle. The first equation senses when the counter and the blanking input
"are equal and the second holds off the TARGET ENABLE until this happens
"and then keeps the enable set until the counter reaches a large range
"value (> 512 us) and then holds it off until the next trigger and blanking
"interval "or until a reset occurs.
BLANK_COMP = (RNG_CNT_LBYTE==BLANK);
TARG_DET_EN.clk = CLOCK_8M;
TARG_DET_EN := RESET & !RNGCOUNT12 & (BLANK_COMP # TARG_DET_EN);
```

```
"Invert BIT_START_L for status. This is done such that if this device fails
"and is interrogated by BIT_START_L, it cannot respond with an active high
"output.
BIT_FAIL_L = !(BIT_START_L);

"The following output is given if more than 2.56 ms goes by between
"triggers. This is 25% more time than any currently known desired trigger
"interval and is intended to be a course timer to detect potential problems
"with the radar.
RADAR_MLFNCTN_L.clk = CLOCK_8M;
RADAR_MLFNCTN_L := !(RESET &
                   ((RNGCOUNT14.fb & RNGCOUNT12.fb) #!RADAR_MLFNCTN_L));

"The following internal nodes set up control signal to allow detection of
"targets in an overlapping set of windows. A logic flow diagram has been
"generated to show this.
SEARCH_A.clk = CLOCK_8M;
SEARCH_B.clk = CLOCK_8M;

            SEARCH_A := RESET & (!RNGCOUNT3.fb # (RNGCOUNT3.fb &
                                ((LSB_COUNTS==5) # (LSB_COUNTS==6) #
                               (LSB_COUNTS==7) # (LSB_COUNTS==0))));

            SEARCH_B := RESET & (RNGCOUNT3.fb # (!RNGCOUNT3.fb &
                              ((LSB_COUNTS==5) # (LSB_COUNTS==6) #
                               (LSB_COUNTS==7) # (LSB_COUNTS==0))));

"Search for targets during the two overlapping windows. If a target is
"found, hold the condition. Reset any conditions prior to the next search.
"Note that target are only set if the time is beyond the blanking window.
TARG_FOUND_A.clk = CLOCK_8M;
TARG_FOUND_B.clk = CLOCK_8M;

TARG_FOUND_A := RESET & !(!SEARCH_A & (LSB_COUNTS==4)) & TARG_DET_EN &
(RAM_BIT.fb>=4) & ((SEARCH_A & !TARG_REG1 & TARG_REG2) # TARG_FOUND_A);

TARG_FOUND_B := RESET & !(!SEARCH_B & (LSB_COUNTS==4)) & TARG_DET_EN &
(RAM_BIT.fb>=4) & ((SEARCH_B & !TARG_REG1 & TARG_REG2) # TARG_FOUND_B);

"Setup the control signal for reading the data in the RAM and latch the
"data into the device only during this time. NOTE THAT READING AND WRITING
"THE RAM ONLY OCCUR FOR THE FIRST 512 US AFTER A TRIGGER.
"This is to ensure that there are no problems with changing the RAM address
"when a write signal is low which could cause incorrect data to be
"gathered.
 RAM_READ_L.clk = CLOCK_8M;
RAM_READ_L := !(RESET & (LSB_COUNTS==0) & !RNGCOUNT12);
RAM_READ_DATA.clk = CLOCK_8M;
RAM_READ_DATA := RESET &
        ((RAM_IN & !RAM_READ_L) # (RAM_READ_DATA & RAM_READ_L));
```

```
RAM_FAIL_L.clk = CLOCK_8M;
RAM_FAIL_L := !(RESET &
                ((RAM_BIT.fb==3) & (RAM_READ_DATA!=0)) # !RAM_FAIL_L));

RAM_WR_L.clk = CLOCK_8M;
RAM_WR_L := !(RESET & !RNGCOUNT12 &
((LSB_COUNTS==4) # (LSB_COUNTS==5) # (LSB_COUNTS==6)));

"Output a chip select coincident with the middle of the write and the data
"output enable and also with the read.
RAM_CS_L.clk = CLOCK_8M;
RAM_CS_L := !(RESET & !RNGCOUNT12 &
                          ((LSB_COUNTS==0) # (LSB_COUNTS==5)));

"Set up the output enable for the RAM data. This is done in this manner to
"try to get the data turned off ASAP to attempt to avoid bus contention.
STATE_5.clk = CLOCK_8M;
STATE_5 :=  (RESET & (LSB_COUNTS==5));

"Set up the outputs for the RAM based on targets found in the
"current pass and the contents of the RAM. RAM_OUT.clk = CLOCK_8M;
RAM_OUT.oe = STATE_5;

WHEN (RAM_BIT.fb!=2) THEN
RAM_OUT0 := RESET & ((RNGCOUNT3 & (LSB_COUNTS==3) & TARG_FOUND_A)
#(!RNGCOUNT3 & (LSB_COUNTS==3) & TARG_FOUND_B) # (RAM_OUT0.fb &
!(LSB_COUNTS==3)));

ELSE
RAM_OUT0 := 0;

WHEN (RAM_BIT.fb!=2) THEN
RAM_OUT1 := RESET & ((RNGCOUNT3 & (LSB_COUNTS==3) & TARG_FOUND_A &
RAM_READ_DATA0) # (!RNGCOUNT3 & (LSB_COUNTS==3) & TARG_FOUND_B &
RAM_READ_DATA0) # (RAM_OUT1.fb & !(LSB_COUNTS==3)));

ELSE
RAM_OUT1 := 0;

"The upper two RAM data bits are reserved for later use.
RAM_OUT2 := 0;
RAM_OUT3 := 0;

"If three passes are selected, look for a target to be in the same bin for
"three passes before it is flagged as a real target. Otherwise look for it
"to be there for two consecutive passes.
TARGET_L.clk = CLOCK_8M;
```

```
WHEN (RANGE_3RD_PASS==1) THEN
TARGET_L :=!(RESET & ((RNGCOUNT3 & (LSB_COUNTS==3) & TARG_FOUND_A &
RAM_READ_DATA0 & RAM_READ_DATA1) # (!RNGCOUNT3 & (LSB_COUNTS==3) &
TARG_FOUND_B & RAM_READ_DATA0 & RAM_READ_DATA1) # (!TARGET_L)));

ELSE
TARGET_L := !(RESET & ((RNGCOUNT3 & (LSB_COUNTS==3) & TARG_FOUND_A &
RAM_READ_DATA0) # (!RNGCOUNT3 & (LSB_COUNTS==3) & TARG_FOUND_B &
RAM_READ_DATA0) # (!TARGET_L)));

"Edged detect the target_l signal and use result to latch the target address
REG_TARGET_L.clk = CLOCK_8M;
REG_TARGET_L := !(RESET & !TARGET_L);

TARGET_ADDR.clk = CLOCK_8M;

WHEN (!TARGET_L & REG_TARGET_L) THEN
TARGET_ADDR := TARGET_DET_ADDR.fb;

ELSE
TARGET_ADDR := TARGET_ADDR.fb;

"Shutdown the laser when any of the following occur:
"Note that the control signals that drive this logic are latching and will
"hold the laser off until a reset occurs.
LASER_EN.clk = CLOCK_8M;
LASER_EN := !((RAM_BIT.fb!=7) # (!TARGET_L) # (!RAM_FAIL_L) #
(!RADAR_MLFNCTN_L) # (!BIT_SHUTDOWN_L));

"Set up state diagram to run the range bin controller through BIT
"prior to starting to gather targets:
RAM_BIT.clk = CLOCK_8M;

STATE_DIAGRAM RAM_BIT

STATE 0: if (RESET==1) then 1 else 0;

STATE 1: if (RESET==0) then 0 else
            if ((!TRIG_REG1 & TRIG_REG2)==1) then 2 else 1;

"Write zeros to RAM
STATE 2: if (RESET==0) then 0 else
            if ((!TRIG_REG1 & TRIG_REG2)==1) then 3 else 2;

"Read data from RAM
STATE 3: if (RESET==0) then 0 else
            if ((!TRIG_REG1 & TRIG_REG2)==1) then 4 else 3;

"Pass 1
STATE 4: if (RESET==0) then 0 else
            if ((!TRIG_REG1 & TRIG_REG2)==1) then 5 else 4;
```

```
"Pass 2
STATE 5: if (RESET==0) then 0 else
           if ((!TRIG_REG1 & TRIG_REG2)==1) then 6 else 5;

"Pass 3
STATE 6: if (RESET==0) then 0 else
           if ((!TRIG_REG1 & TRIG_REG2)==1) then 7 else 6;

"Enable LASER and STAY HERE UNTIL RESET
STATE 7: if (RESET==0) then 0 else 7;


END rng_ctrl;
```

Appendix E

C Code for Generation of Threshold Control Lookup PROM

```
/*************************************************************************

        Source File:                    lsr_threshold_rom.c
        Intel Hex File(s):              THRESHOLD.hex
        Designer:                       S. Boone
        Unprogrammed P/N:               AM27291A

        Revision history:

        Date              By      Rev     Comments
        ----              ---     -----     --------------------
        22-AUG-1994       spb     0.0     Creation Date.
        24-JAN-1995       spb     0.1     Changed to final threshold values

        The make command is: cc -o lsr_threshold_rom lsr_threshold_rom.c \
                                   make_intel_hex_file.c

        The command to execute is: lsr_threshold_rom
*************************************************************************

                                Description

This file create the Intel-Hex ROM code for the THRESHOLD ROM on the laser
safety radar controller. It outputs a code to be D/A converted to allow for
a changing threshold value for the VD signal from the R72. This
implementation will cause the circuit to output discrete values from ~1.90
volts to ~2.10 volts.

*/
#define PROM_LENGTH 2048
#define PROM_WIDTH 8

main()
  {
  unsigned int data[PROM_LENGTH];
  unsigned int index;
  unsigned char prom[PROM_LENGTH];

  int i;
  index=0;


/* Set initial value  */
  data[0] = (190 & 0xff);
```

```
/* Next 5 locations ( 1 - 5) */
  for (i=0; i<5; i++)
    {
    data[index + 1] = (190 & 0xff);
    index++;
    }

/* Second 18 locations (6 - 23) */
  for (i=0; i<18; i++)
    {
    data[index] = (195 & 0xff);
    index++;
    }

/* Next 22 locations (24 - 45) */
  for (i=0; i<22; i++)
    {
    data[index] = (200 & 0xff);
    index++;
    }

/* Next 28 locations (46 - 73) */
  for (i=0; i<28; i++)
    {
    data[index] = (205 & 0xff);
    index++;
    }

/* Next 438 locations (74 - 511) */
  for (i=0; i<438; i++)
    {
    data[index] = (210 & 0xff);
    index++;
    }

/* Next 512 locations (512 - 1023) */
  for (i=0; i<512; i++)
    {
    data[index] = (210 & 0xff);
    index++;
    }

/* Last1024 locations (1024 - 2047) */
 for (i=0; i<1024; i++)
    {
    data[index] = (210 & 0xff);
    index++;
    }

 printf ("\n\tfinal index is %i",(index-1));
```

```
/* Create array of characters for intel hex routine */
 for (i=0; i<PROM_LENGTH; i++)
   {
   prom[i] = (data[i] & 0x00ff);
   }

 printf ("\n\tData array generated\n");

 make_intel_hex_file(prom,
                 PROM_LENGTH,
                 0,
                 0x00,
                 "THRESHOLD.hex");
  }
```

```
/*
   make_intel_hex_file.c                              Jeffrey S. Weber 9/92

   This code creates an Intel Hex PROM file from a data array.

   Rev. 1.0  12/4/92   jsw
       fixed it so that hex numbers are uppercase.
       fixed it so that the checksum is 2s complement.

   Rev 1.1   4/13/93 spb
       changed to allow the output of one extra location for a
       prom containing a control byte.

*/

#include <stdio.h>
#define MAX_ADDRESS (64 * 1024)
#define NUM_ADDRESS_PER_RECORD (16)
#define EXT_ADDRESS_RECORD ":020000040001F9\n"
#define INTEL_EOF_RECORD ":00000001FF"
#define TRUE 1
#define FALSE 0
int make_intel_hex_file( array, length, ctrl_byte, ctrl_byte_val, outfile)
  unsigned char *array, ctrl_byte_val;
  int length;
  short ctrl_byte;
  char *outfile;
  {
  FILE *fp;
  int address=0, i ;
  unsigned char checksum;

  if (length > MAX_ADDRESS)
    {
    printf("make_intel_hex_file: array too large\n");
    return(1);
    }

  if ( (fp = fopen(outfile, "w+")) == (FILE *) NULL )
    {
    printf("make_intel_hex_file: cannot open file %s\n", outfile);
    return(1);
    }

 while (address < length)
    {
    fprintf(fp,":10%04X00",address);
    checksum = 0x10;
    checksum += (address & 0x00ff);
    checksum += (address & 0xff00) >> 8;
    checksum += 0x00;
    for (i=0; i<NUM_ADDRESS_PER_RECORD; i++)
```

```
    {
    fprintf(fp,"%02X", array[address]);
    checksum += array[address];
    address++;
    }
  checksum = 0xff & ((~checksum) + 1);      /* make 2s complement */
  fprintf(fp,"%02X\n", checksum);
  }

if (ctrl_byte==TRUE)
  {
  fprintf(fp, EXT_ADDRESS_RECORD); /* Output Extended address */
  fprintf(fp,":01000000");
  fprintf(fp,"%02X", ctrl_byte_val);
  checksum = 0x01;
  checksum += ctrl_byte_val;
  checksum = 0xff & ((~checksum) + 1);      /* make 2s complement */
  fprintf(fp,"%02X\n", checksum);
  }
fprintf(fp, INTEL_EOF_RECORD);
fclose(fp);
printf("make_intel_hex_file: %s written\n", outfile);

return(0);
}
```

Appendix F

MATLAB Code for Thesis Calculations

```
%MATLAB Program
% Calculate the Half Power Range for a Raytheon R72
clear all
hold off
%Define radar system parameters
Pt = 10000.;
Gain = 1000.; % 30 db

% Define feet per meter conversion: (do not convert to feet)
%fpm = 100/(2.54*12);
 fpm = 1;

% Define R72 receiver noise floor
noise=[-103.23];

% Minimum detectable signal above noise floor
delta = [6.,9.,12.,15.];

% Convert power in dBm to watts
 smin = (10.^((noise+delta)/10.))/1000.;

% Define radar frequency with actual range:

xfreq = 9410.e6;
c=3.e8;
lambda = c/xfreq;

% Get values for Max range vs. radar cross
% section (sigma)

sigma = [1.:100.];

 rmax  = ((Pt*Gain^2 *(lambda)^2 * sigma)/((4*pi)^3 * smin(1))).^0.25;

 %print results to screen:
 rmax(1)*fpm
 rmax(5)*fpm
 rmax(10)*fpm
 rmax(20)*fpm
 rmax(40)*fpm
 rmax(100)*fpm

 %plot on log/log scale for linearity and convert from meters to feet
 loglog(rmax*fpm);
 hold on
 loglog(1,rmax(1)*fpm,'+');
 loglog(5,rmax(5)*fpm,'+');
```

```
loglog(10,rmax(10)*fpm,'+');
loglog(20,rmax(20)*fpm,'+');
loglog(40,rmax(40)*fpm,'+');
loglog(100,rmax(100)*fpm,'+');

rmax  = ((Pt*Gain^2 *(lambda)^2 * sigma)/((4*pi)^3 * smin(2))).^0.25;

%print results to screen:
rmax(1)*fpm
rmax(5)*fpm
rmax(10)*fpm
rmax(20)*fpm
rmax(40)*fpm
rmax(100)*fpm

%plot on log/log scale for linearity and convert from meters to feet
loglog(rmax*fpm);
loglog(1,rmax(1)*fpm,'o');
loglog(5,rmax(5)*fpm,'o');
loglog(10,rmax(10)*fpm,'o');
loglog(20,rmax(20)*fpm,'o');
loglog(40,rmax(40)*fpm,'o');
loglog(100,rmax(100)*fpm,'o');

rmax  = ((Pt*Gain^2 *(lambda)^2 * sigma)/((4*pi)^3 * smin(3))).^0.25;
%print results to screen:
rmax(1)*fpm
rmax(5)*fpm
rmax(10)*fpm
rmax(20)*fpm
rmax(40)*fpm
rmax(100)*fpm

%plot on log/log scale for linearity and convert from meters to feet
loglog(rmax*fpm);
hold on
loglog(1,rmax(1)*fpm,'*');
loglog(5,rmax(5)*fpm,'*');
loglog(10,rmax(10)*fpm,'*');
loglog(20,rmax(20)*fpm,'*');
loglog(40,rmax(40)*fpm,'*');
loglog(100,rmax(100)*fpm,'*');

rmax  = ((Pt*Gain^2 *(lambda)^2 * sigma)/((4*pi)^3 * smin(4))).^0.25;

%print results to screen:
rmax(1)*fpm
rmax(5)*fpm
rmax(10)*fpm
rmax(20)*fpm
rmax(40)*fpm
rmax(100)*fpm
```

```
 %plot on log/log scale for linearity and convert from meters to feet
 loglog(rmax*fpm);
 loglog(1,rmax(1)*fpm,'x');
 loglog(5,rmax(5)*fpm,'x');
 loglog(10,rmax(10)*fpm,'x');
 loglog(20,rmax(20)*fpm,'x');
 loglog(40,rmax(40)*fpm,'x');
 loglog(100,rmax(100)*fpm,'x');

% axis ([1,100,7e3,6e4]);
 title('Maximum Range versus Radar Cross Section ');
 text(1.2,5e4,'Signal Detection Level Above Noise:');
 text(1.4,4.5e4,'6 dB (+)');
 text(1.4,4.1e4,'9 dB (o)');
 text(1.3,3.7e4,'12 dB (*)');
 text(1.3,3.3e4,'15 dB (x)');
 xlabel('Radar Cross Section (square meters)');
 ylabel('Maximum Range (meters)');

hold off
```

```
%MATLAB Program
% Calculate the Half Power Range for a Raytheon R72

clear all
hold off
%Define radar system parameters
Pt = 10000.;
Gain = 501.1872336; % 27 db

% Define feet per meter conversion (do not convert here):
%fpm = 100/(2.54*12);
fpm = 1;

% Define R72 receiver noise floor
noise=[-103.23];

% Minimum detectable signal above noise floor
delta = [6.,9.,12.,15.];

% Convert power in dBm to watts
 smin = (10.^((noise+delta)/10.))/1000.;

% Define radar frequency with actual range:

xfreq = 9410.e6;
c=3.e8;
lambda = c/xfreq;

% Get values for Max range vs. radar cross
% section (sigma)

sigma = [1.:100.];

 rmax  = ((Pt*Gain^2 *(lambda)^2 * sigma)/((4*pi)^3 * smin(1))).^0.25;

 %print results to screen:
 rmax(1)*fpm
 rmax(5)*fpm
 rmax(10)*fpm
 rmax(20)*fpm
 rmax(40)*fpm
 rmax(100)*fpm

 %plot on log/log scale for linearity and convert from meters to feet
 loglog(rmax*fpm);
 hold on
 loglog(1,rmax(1)*fpm,'+');
 loglog(5,rmax(5)*fpm,'+');
 loglog(10,rmax(10)*fpm,'+');
 loglog(20,rmax(20)*fpm,'+');
 loglog(40,rmax(40)*fpm,'+');
 loglog(100,rmax(100)*fpm,'+');
```

```
rmax  = ((Pt*Gain^2 *(lambda)^2 * sigma)/((4*pi)^3 * smin(2))).^0.25;

%print results to screen:
rmax(1)*fpm
rmax(5)*fpm
rmax(10)*fpm
rmax(20)*fpm
rmax(40)*fpm
rmax(100)*fpm

%plot on log/log scale for linearity and convert from meters to feet
loglog(rmax*fpm);
loglog(1,rmax(1)*fpm,'o');
loglog(5,rmax(5)*fpm,'o');
loglog(10,rmax(10)*fpm,'o');
loglog(20,rmax(20)*fpm,'o');
loglog(40,rmax(40)*fpm,'o');
loglog(100,rmax(100)*fpm,'o');

rmax  = ((Pt*Gain^2 *(lambda)^2 * sigma)/((4*pi)^3 * smin(3))).^0.25;
%print results to screen:
rmax(1)*fpm
rmax(5)*fpm
rmax(10)*fpm
rmax(20)*fpm
rmax(40)*fpm
rmax(100)*fpm

%plot on log/log scale for linearity and convert from meters to feet
loglog(rmax*fpm);
hold on
loglog(1,rmax(1)*fpm,'*');
loglog(5,rmax(5)*fpm,'*');
loglog(10,rmax(10)*fpm,'*');
loglog(20,rmax(20)*fpm,'*');
loglog(40,rmax(40)*fpm,'*');
loglog(100,rmax(100)*fpm,'*');

rmax  = ((Pt*Gain^2 *(lambda)^2 * sigma)/((4*pi)^3 * smin(4))).^0.25;

%print results to screen:
rmax(1)*fpm
rmax(5)*fpm
rmax(10)*fpm
rmax(20)*fpm
rmax(40)*fpm
rmax(100)*fpm

%plot on log/log scale for linearity and convert from meters to feet
loglog(rmax*fpm);
loglog(1,rmax(1)*fpm,'x');
```

```
loglog(5,rmax(5)*fpm,'x');
loglog(10,rmax(10)*fpm,'x');
loglog(20,rmax(20)*fpm,'x');
loglog(40,rmax(40)*fpm,'x');
loglog(100,rmax(100)*fpm,'x');

%axis ([1,100,3e3,4e4]);
title('Half Power Range versus Radar Cross Section ');
text(1.2,3e4,'Signal Detection Level Above Noise:');
text(1.4,2.6e4,'6 dB (+)');
text(1.4,2.3e4,'9 dB (o)');
text(1.3,2.0e4,'12 dB (*)');
text(1.3,1.7e4,'15 dB (x)');
xlabel('Radar Cross Section (square meters)');
ylabel('Half Power Range (meters)');

hold off
```

```
% MATLAB Program
% Probability of Detection
pc=0.99999;
pd = [0.1:0.05:0.9];
n=log10(1-pc)./(log10(1-pd));
plot(n,pd);
xlabel('Number of Pulses');
ylabel('Detection Probability of Single Pulse ');
text(18,0.7,'Cumulative Probability of Detect = 0.99999');
```