TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA/TX-87/50+309-1F		

4. Title and Subtitle

THE DESIGN OF AN AUTOMATED TRAFFIC COUNTING SYSTEM WITH TURNING MOVEMENT FLOW ANALYSIS

5. Report Date
November 1986

6. Performing Organization Code

7. Author(s)
Ann M. Mechler, Randy B. Machemehl, and Clyde E. Lee

8. Performing Organization Report No.
Research Report 309-1F

9. Performing Organization Name and Address

Center for Transportation Research
The University of Texas at Austin
Austin, Texas 78712-1075

10. Work Unit No.

11. Contract or Grant No.
Research Study 3-18-81-309

12. Sponsoring Agency Name and Address

Texas State Department of Highways and Public Transportation; Transportation Planning Division
P. O. Box 5051
Austin, Texas 78763-5051

13. Type of Report and Period Covered
Final

14. Sponsoring Agency Code

15. Supplementary Notes
Study conducted in cooperation with the U. S. Department of Transportation, Federal Highway Administration. Research Study Title: "Automatic Turning Movement Count System for Signalized Intersections"

16. Abstract

The feasibility of developing an automated system for making vehicular traffic counts for at-grade intersections was investigated. Various conceptual approaches were explored; however, all calculation methods which attempted to deduce traffic-turning volumes directly from detector information were found to produce fewer than the required number of independent equations.

A new technique based upon real-time forecasts of the time required for making right-turn movements is proposed. The technique utilizes the right-turn forecasting technique as a means of supplementing information collected from a system of detectors on inbound and outbound lanes, and thereby allows calculation of all traffic movements through the intersection. The accuracy of the method is improved by sensing the "state" of the intersection signal controller. The signal-controller status permits direct calculation of all turning movements except right-turns made during green-signal indications. The proposed counting methodology was implemented through the use of off-the-shelf hardware consisting of an open-architecture microcomputer and a TTL-compatible data-acquisition board. Although the technique is independent of detector type, use of a vehicle, rather than an axle detection device, is strongly preferred.

As an extension of the primary research effort, several right-turn related problems were examined. Equivalence relationships for right-turn and straight-through traffic movements within the context of volume warrants for signal installation were developed.

17. Key Words	18. Distribution Statement
right turns, automatic turning movement counting, equivalence factors, right-turn bays, right-turn lanes, detection systems	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	114	

Form DOT F 1700.7 (8-69)

THE DESIGN OF AN AUTOMATED TRAFFIC COUNTING SYSTEM WITH TURNING MOVEMENT FLOW ANALYSIS

by

Ann M. Mechler
Randy B. Machemehl
Clyde E. Lee

Research Report Number 309-1F

Automatic Turning Movement Count System
for Signalized Intersections

Research Project 3-18-81-309

conducted for

Texas
State Department of Highways and Public Transportation

in cooperation with the
U. S. Department of Highways and Public Transportation
Federal Highway Administration

by the

CENTER FOR TRANSPORTATION RESEARCH
THE UNIVERSITY OF TEXAS AT AUSTIN

November 1986

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the Federal Highway Administration. This report does not constitute a standard, specification, or regulation.

There was no invention or discovery conceived or first actually reduced to practice in the course of or under this contract, including any art, method, process, machine, manufacture, design or composition of matter, or any new and useful improvement thereof, or any variety of plant which is or may be patentable under the patent laws of the United States of America or any foreign country.

ABSTRACT

The feasibility of developing an automated system for making vehicular traffic counts for at-grade intersections was investigated. Various conceptual approaches were explored; however, all calculation methods which attempted to deduce turning-traffic volumes directly from detector information were found to produce fewer than the required number of independent equations.

A new technique based upon real-time forecasts of the time required for making right-turn movements is proposed. The technique utilizes the right-turn forecasting technique as a means of supplementing information collected from a system of detectors on inbound and outbound lanes, and thereby allows calculation of all traffic movements through the intersection. The accuracy of the method is improved by sensing the "state" of the intersection signal controller. The signal-controller status permits direct calculation of all turning movements except right-turns made during green-signal indications.

The proposed counting methodology was implemented through the use of off-the-shelf hardware consisting of an open-architecture microcomputer and a TTL-compatible data-acquisition board. Although the technique is independent of detector type, use of a vehicle, rather than an axle detection device is strongly preferred.

As an extension of the primary research effort, several right-turn related problems were examined. Equivalence relationships for right-turn and straight-through traffic movements within the context of volume warrants for signal installation were developed. Guidelines for the use of right-turn bays and lanes adjacent to intersections are also suggested.

KEY WORDS: right turns, automatic turning movement counting, equivalence factors, right-turn bays, right-turn lanes, detection systems

SUMMARY

Several technical approaches to automating the, usually manual, process of counting vehicular traffic movements at signalized intersections have been examined. An automated counting technique which uses detectors to count traffic entering and leaving an intersection, forecasts the time used for making right-turns, and monitors the green-signal indications is proposed. The time forecasts for right-turns, when coupled with direct counts of vehicles entering and leaving the intersection, provide the basic elements of data that are needed to solve a system of equations which account for all traffic movements through the intersection.

The technique has been implemented using an off-the-shelf microcomputer and interface device along with detectors. Several detector types were tested, but the most desirable was found to be an infra-red light beam reflex device which can be used as a vehicle rather than an axle detector.

The study efforts were extended to encompass two operational aspects of right-turn operations through intersections. Equivalence relationships for right-turn and straight-through movements from an approach were developed in terms of relative maximum flow rates. These relationships are intended for use in traffic volume warrants for signal installation. Guidelines for the use of right-turn bays and lanes adjacent to signalized intersections were developed. These guidelines are based upon maximum queue lengths which will develop in through-traffic lanes when red-signal indications might extend up to 80 seconds.

IMPLEMENTATION STATEMENT

A technique for automating the process of counting vehicular traffic movements through signalized, intersections is described herein. The computing algorithm and an off-the-shelf hardware system that are needed for field implementation of the technique are detailed. Guidelines for the use of right-turn bays and lanes at intersections are suggested, and equivalency relationships for right-turning and straight-through traffic movements in relation to traffic volume warrants for signal installation are presented. These techniques and guidelines may be applied by engineers in both the public and private sectors for the collection and interpretation of data that are necessary for signal timing and for the design and redesign of intersections.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Accurate traffic flow counts are needed in many situations. In upgrading, channelizing, or signalizing an intersection the flow counts must include estimates of the proportions of traffic turning left and right from each approach of the intersection. In addition, it is usually desirable to obtain this information as quickly and inexpensively as possible. Approach flows are easily and inexpensively determined by automatic counting machines; however, the collection of good turning movement counts has traditionally been problematic. This paper presents the basis for the design of an automated counting device that also estimates turning flows for all intersection approaches.

There are several reasons for the difficulty in obtaining turning movement estimates. The accuracy of automatic counting depends to a large extent on the type of detector used. Loop detectors, pneumatic pressure tubes, and tape switches each have small amounts of error associated with them. This error is generally acceptable for determining approach flows. If more detectors are placed at the intersection to estimate the turning movement flows, the quality of the detecting device, as well as the ease of installation, becomes important. Even if these difficulties are overcome, enough detectors cannot be placed at an unchannelized intersection to determine the complete flow matrix. For example, eight independent equations for flow can be written if one detector is placed on each set of inbound and outbound lanes of a four-leg intersection, but 12 unknowns exist (three movements from each set of inbound lanes on four legs) if all movements are allowed. This problem is discussed in detail in the next chapter.

If the number of vehicles making one type of movement can be estimated indirectly on each approach, the other flow counts can be calculated directly with the eight equations. A method for estimating right turn counts is presented in this paper. The method is based on the idea that the time for a right-turning vehicle to travel between detectors placed close to the intersection across both the inbound and outbound lanes on each street can be predicted if a small margin of error is allowed. In addition, right turns on red can be estimated when the signalization pattern is known. Thus, the through counts and left turn counts can be determined from the total counts at the eight detectors. A proposed design for an interface between the detectors and a microcomputer to be used for computing the counts is also included.

Since right turning traffic became a central issue in the turning movement analyses, the study was expanded to deal with several questions related to right-turn treatments. Analyses and research results regarding this aspect of the study are present in Chapter 5.

CHAPTER 2. PROBLEM DESCRIPTION AND BACKGROUND

In many intersection studies it is necessary to have reliable traffic counts for all permitted vehicle movements. Some terminology is introduced here that is used in describing the proposed method of determining such counts.

The number of vehicles traveling from Leg i to Leg j of the intersection is denoted as N_{ij}. Traffic originating at Leg i is then $O_i = \sum_{\text{all } i} N_{ij}$, and traffic departing on Leg j is $D_j = \sum_{\text{all } i} N_{ij}$. The traffic flows can be represented in matrix form as shown in Fig 2-1. In this paper it will be assumed that U-turns are not permitted, thus the diagonal of the flow matrix will always be zero.

FLOW DETERMINATION USING TRAFFIC DETECTORS

Completely Channelized Intersections

The flow matrix for completely channelized intersections can be easily defined when 12 detectors are placed in the intersection to determine the 12 unknowns. Figure 2-2 illustrates some possible detector placements. The detectors are identified by dashed lines and upper-case letters. A corresponding lower-case letter represents the numerical count at a detector. With the detector placement on Leg 1, the movement counts can be determined directly at each detector. On Leg 2, the left and right-turn counts can be taken directly from D and F, respectively. The straight-through flow must be calculated. Since $O_2 = \sum_{j=1}^{4} N_{2j}$, then $e = f + d + n_{24}$. Therefore, $n_{24} = e - f - d$. All turning movement flows can be solved for using the detectors on Legs 3 and 4 as well. Furthermore, one of the detectors may be replaced by a detector on the departing lanes, but the total number of detectors must still equal the number of unknowns.

If an automatic counter is attached to the signal controller such that separate counting banks can be maintained for detections occurring in each of the different phases of each cycle, then the number of detectors can be reduced. In the case of a fully channelized intersection operating on two phase control, the number of detectors can be reduced to eight. Figure 2-3 shows one detector configuration and the resulting flow matrix. The subscripts 1 and 2 on the flow counts represent Phase 1 which services Legs 1 and 3 and Phase 2 which services Legs 2 and 4. In this example, all of the traffic counts are obtained directly. Furthermore, it does not matter whether the left turns are channelized since no detector is needed in the left-turn bay or lane. Since with the help of the signal controller the detector on the departing lanes is doing the work of two detectors, the problem of estimating the turning flows is simplified. Four-phase signalization with protected left-turn phases reduces the number of required detectors to four.

TO / FROM	1	2	3	4	
1		N_{12}	N_{13}	N_{14}	O_1
2	N_{21}		N_{23}	N_{24}	O_2
3	N_{31}	N_{32}		N_{34}	O_3
4	N_{41}	N_{42}	N_{43}		O_4
	D_1	D_2	D_3	D_4	

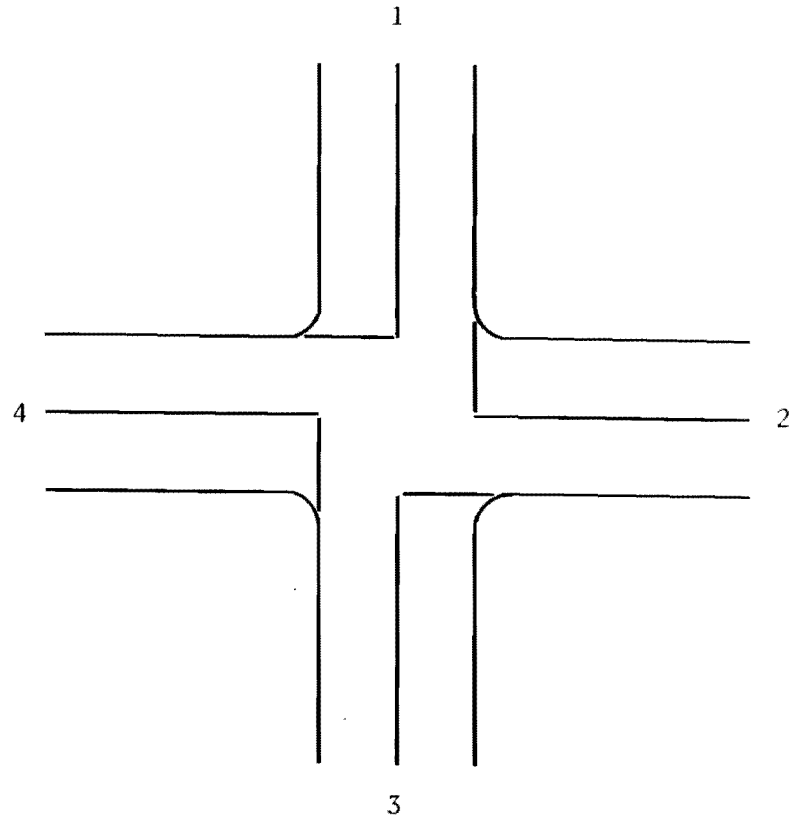
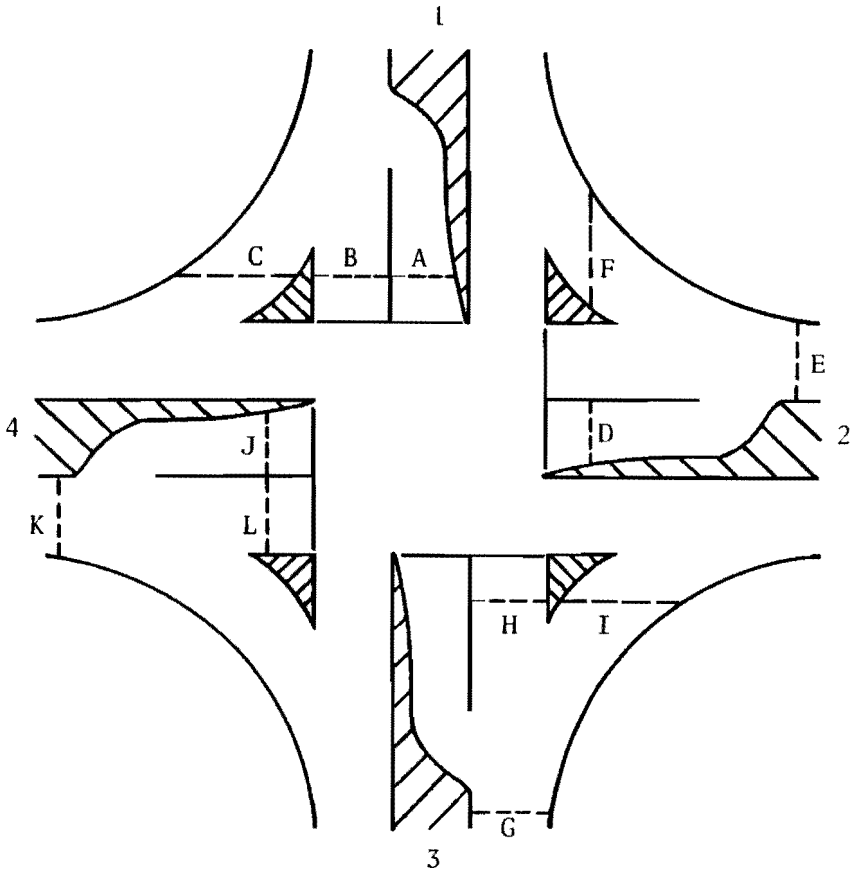
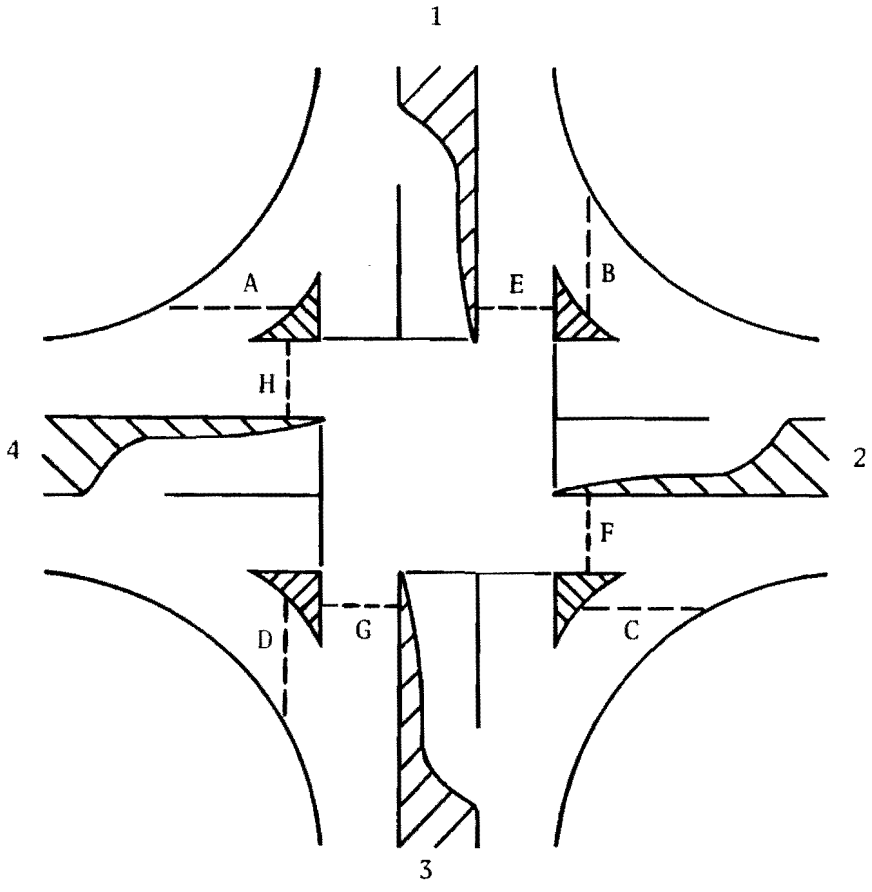


Figure 2-1. Flow matrix and intersection.



TO / FROM	1	2	3	4	
1		a	b	c	o_1
2	f		d	n_{24}	e
3	h	i		n_{34}	g
4	j	l	n_{43}		k
	D_1	D_2	D_3	D_4	

Figure 2-2. Completely channelized intersection and flow matrix.



TO / FROM	1	2	3	4	
1		f_1	g_1	a	o_1
2	b		g_2	h_2	o_2
3	e_1	c		h_1	o_3
4	e_2	f_2	d		o_4
	D_1	D_2	D_3	D_4	

D_i = total traffic volume departing intersection having entered from Approach i

O_j = total traffic entering intersection on Approach j

Figure 2-3. Fully channelized intersection and flow matrix with counts separated by signal phases. Phase 1 allows movement from Legs 1 and 3. Phase 2 services Legs 2 and 4.

Partially Channelized Intersections

As mentioned in the previous section, the turning movement counts for an intersection with only channelized right turns can be fully determined with eightdetectors and the signal controller connected to an automatic counter with twelve storage banks. It can be shown that two other types of partially channelized intersections can be analyzed in a similar way. Figure 2-4 illustrates three cases and their flow matrices. Figure 2-4a is the same as Figure 3 without left turn bays. Figure 2-4b shows an intersection with left turn bays and no channelized right turns. The flow matrix indicates that the right turns cannot be counted directly. They can easily be solved for since $D_j = \sum_{\text{all } i} N_{ij}$. For example,

$$D_1 = e = n_{21} + e_1 + d$$

Therefore, $n_{21} = e - e_1 - d$. The other right turn flows can be obtained similarly. Figure 2-4c is an intersection with right-turn channelization on two opposing legs, and left-turn bays on the remaining two legs. The turn counts for this case can also be determined from the flow matrix. In summary, when signal controller indications can be utilized in conjunction with traffic counts, the flows for all movements can be determined (1) if all right turns or all left turns are channelized, or (2) if right turns on one pair of opposing approaches and left turns on the other opposing approaches are channelized. Additional channelization does not affect those results. The solution proposed in Figs 2-4a through 2-4c is based upon a signal control with a minimum of two phases. An equally satisfactory solution can be devised for multiphase control.

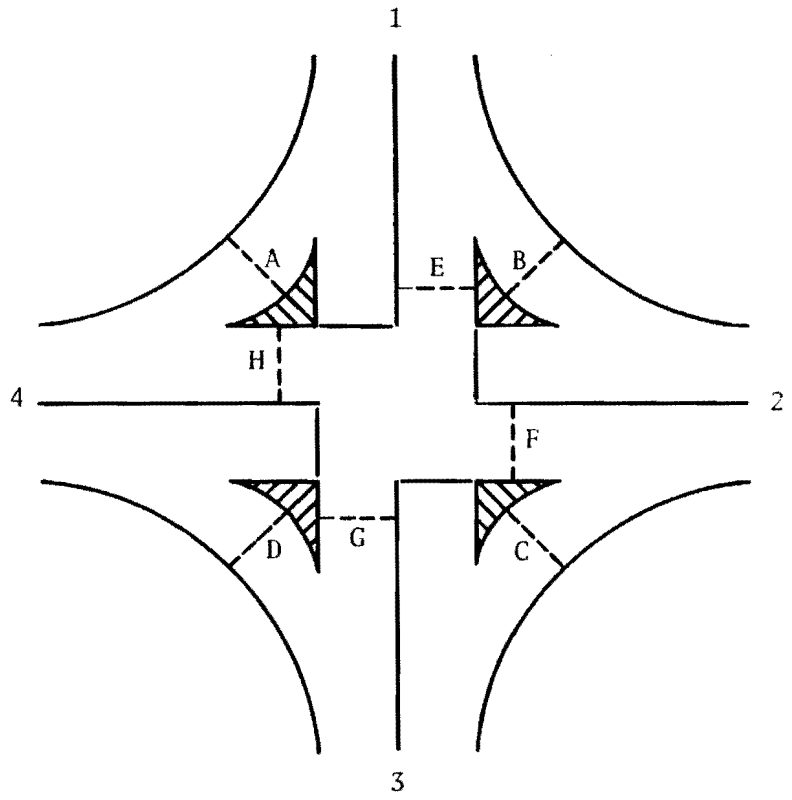
Unchannelized Intersections

It has been shown that for fully-channelized, and for some partially-channelized intersections, all traffic flows can be estimated if enough detectors are placed across the lanes near the intersection and/or the signal controller is tied to the counter. For unchannelized intersections, however, this is not true. A simple two-by-two intersection with detectors across each inbound and outbound lane is shown with its flow matrix in Fig 2-5. In this case, the straight-through traffic counts can be obtained directly from the opposing outbound lane detectors when the counts are recorded separately for each phase. This is only true when a simplifying assumption is made prohibiting right turns on red. Right turns on red will be discussed in detail later in this paper. The equations for the straight through flows can be written:

$$n_{13} = g \tag{2-1}$$

$$n_{24} = h_2 \tag{2-2}$$

$$n_{31} = e_1 \tag{2-3}$$

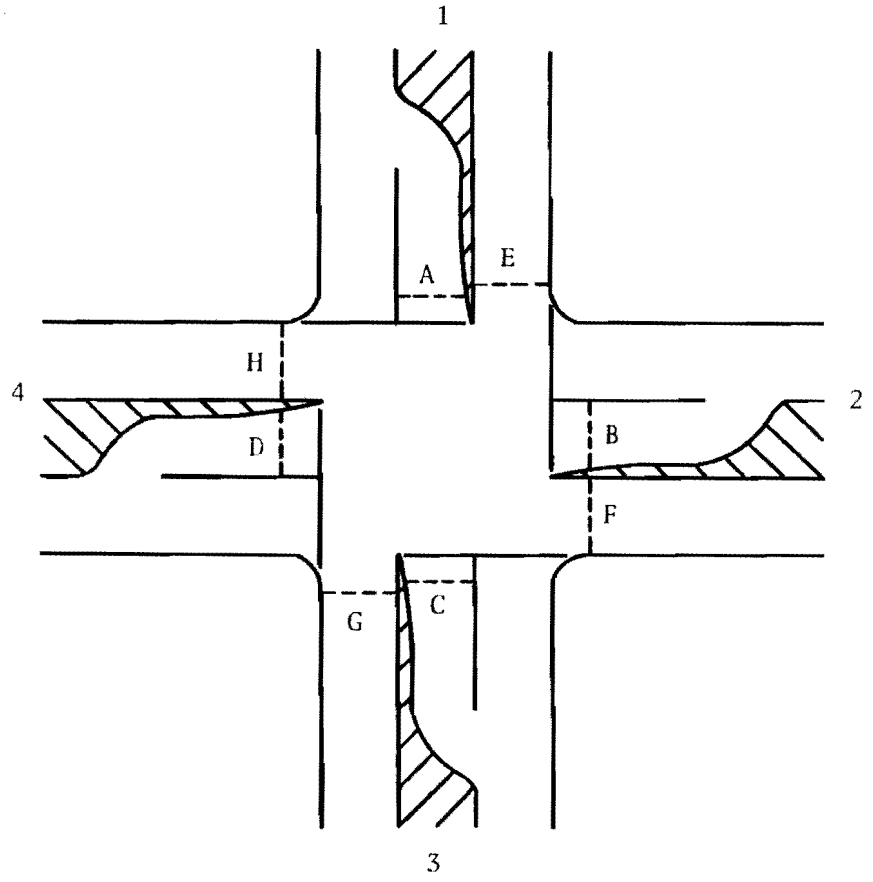


TO / FROM	1	2	3	4	
1		f_1	g_1	a	o_1
2	b		g_2	h_2	o_2
3	e_1	c		h_1	o_3
4	e_1	f_2	d		o_4
	D_1	D_2	D_3	D_4	

D_i = total traffic volume departing intersection having entered from Approach i

O_j = total traffic entering intersection on Approach j

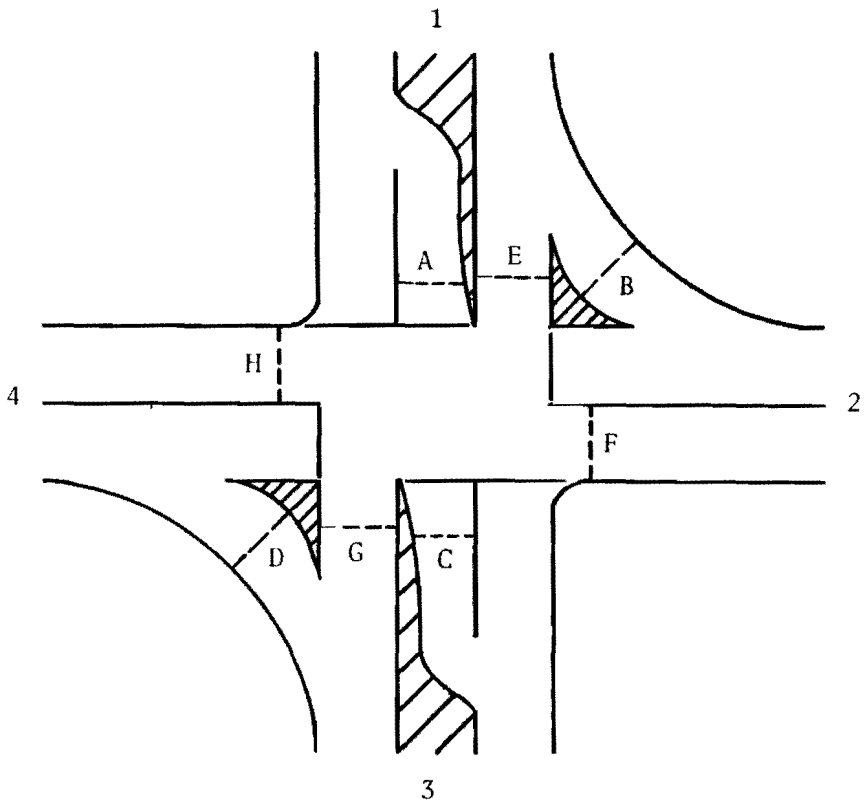
Figure 2-4a. Intersection with channelized only for right-turn movements and flow matrix.



FROM \ TO	1	2	3	4	
1		a	g_1	n_{14}	o_1
2	n_{21}		b	h_2	o_2
3	e_1	n_{32}		c	o_3
4	d	f_2	n_{13}		o_4
	e	f	g	h	

O_j = total approach entering intersection on Approach j

Figure 2-4b. Intersection with channelization only for left-turn movements and flow matrix.

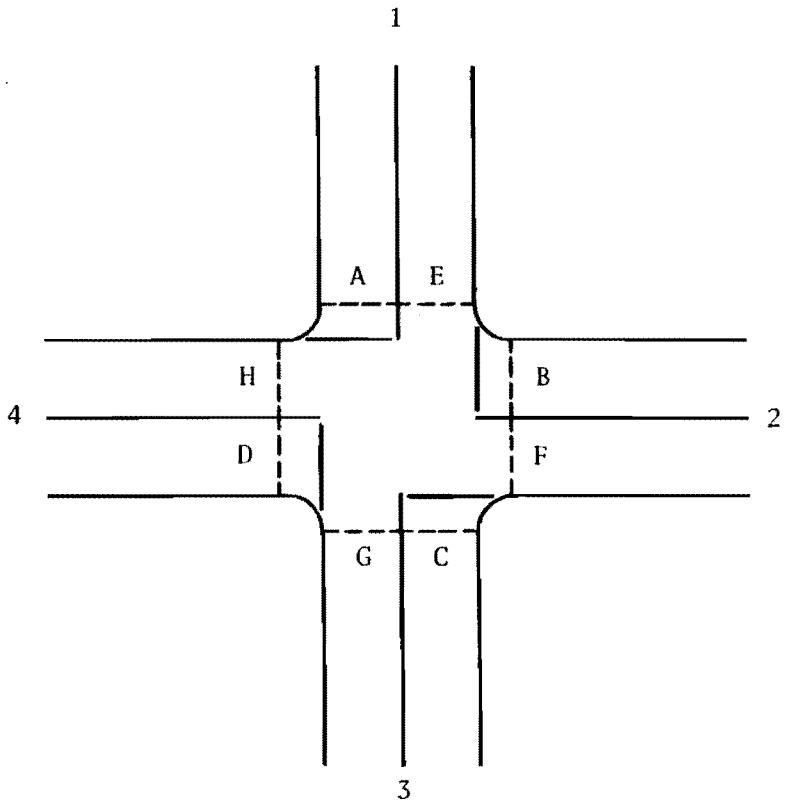


FROM \ TO	1	2	3	4	
1		a	g_1	n_{14}	o_1
2	b		g_2	h_2	o_2
3	e_1	n_{32}		c	o_3
4	e	f_2	d		o_4
	D_1	f	D_3	h	

D_i = total traffic volume departing intersection having entered from Approach i

O_j = total traffic entering intersection on Approach j

Figure 2-4c. Intersection with channelized right turns on two approaches and channelized left turns on two approaches and flow matrix.



TO / FROM	1	2	3	4	
1		n_{12}	g_1	n_{14}	a
2	n_{21}		n_{23}	h_2	b
3	i_1	n_{32}		n_{34}	c
4	n_{41}	f_2	n_{43}		d
	e	f	g	h	

1 and 3 are Serviced by Signal Phase 1

2 and 4 are Serviced by Signal Phase 2

Figure 2-5. Unchannelized intersection and flow matrix.

$$n_{42} = f_2 \tag{2-4}$$

It is also known that $O_i = \sum_{\text{all } j} N_{ij}$. So four more equations can be written:

$$n_{12} + n_{14} = a - g_1 \tag{2-5}$$

$$n_{21} + n_{23} = b - h_2 \tag{2-6}$$

$$n_{32} + n_{34} = c - e_1 \tag{2-7}$$

$$n_{41} + n_{43} = d - f_2 \tag{2-8}$$

Four additional equations can be written from the expression $D_j = \sum_{\text{all } i} N_{ij}$, but these equations are not independent from Equations (2-5) through (2-8). As a result, there are only eight independent equations with which to solve for twelve unknowns. Thus, the problem has no direct solution.

OTHER METHODS OF OBTAINING FLOWS

Other methods have been used to obtain the flow matrix, since the counts cannot be made automatically. The most common way is to have individuals count the turning traffic manually. This is usually too expensive (and unpleasant) to be done frequently. Furthermore, the counts become less accurate over time due to fatigue. Consequently, previously-determined turn proportions are sometimes used to adjust or apportion new total approach counts. This method generally yields poor results, particularly in urban areas where new or relocated businesses can affect traffic flows tremendously over even short periods.

Several estimation methods have been suggested to obtain movement flows from inbound and outbound approach counts [Refs 1-5]. A method proposed by Hauer, et al [Ref 5] uses prior knowledge of the turning proportions to obtain an estimate of the new flows by iterative computations. The method gives reasonable results and has been incorporated into a FORTRAN program. When no prior knowledge of the turning proportions is available, however, the estimates may not be sufficiently accurate, and manual counting is necessary.

THE PROPOSED METHOD OF OBTAINING FLOWS

To eliminate the cost and error of manual counting, it is desirable to develop an automatic counter that can estimate turning movement flows at intersections. As discussed previously, enough detectors can be placed near a channelized intersection to determine all movement counts. This is also true for some partially-channelized intersections, particularly those where all right turns or left turns are channelized and can be counted

directly. In the case of an unchannelized intersection the turn movement flows cannot be determined since neither of the turn flows can be counted directly.

The movement times for vehicles traversing an intersection can provide additional information about the flows. Left-turn movements cannot be reliably estimated using movement durations because left-turning cars might be "trapped" in the intersection while waiting for gaps in the opposing traffic stream. The time a vehicle is trapped can vary widely, thus causing much dispersion in left-turn times. Right-turns, on the other hand, are generally unimpeded after the driver has chosen an acceptable turning opportunity; thus, the times to complete the movement are relatively consistent. Consequently, detection times can be used to estimate right-turn flows, which then enables the remaining flows to be calculated. The flow analysis becomes very similar to that for an intersection with only channelized right turns, as discussed previously. This method of analysis is discussed in detail in the following chapter.

CHAPTER 3. SOLUTION DEVELOPMENT

The time required for a vehicle to complete a movement depends on the length of the path, the average speed, the acceleration and/or deceleration, and the presence or absence of conflicting traffic. The paths for the three possible movements from an approach are illustrated in Fig 3-1. Each path begins at the detector on the inbound lanes and ends at a detector on the outbound lanes. If all outbound lane detectors are placed approximately the same distance from the intersection (detector placement is discussed later), the right-turn path is the shortest. The straight and left-turn paths are considerably longer. Fluctuations in speed due to acceleration or deceleration have more effect on the time to complete the movement when the path is long. Thus, speed fluctuations affect right-turn times the least.

The maximum speed that a turning vehicle can safely travel is limited by the radius of the turn. Right turning vehicles must travel at lower speeds than left turners because the right-turn radius is short. At the lower maximum speed, the right-turn movement has a smaller range of possible speeds. Furthermore, right-turn vehicles have priority of movement during their green phase, so their movements are uninterrupted. Left-turn vehicles, however, may have to yield to conflicting right-turn and through traffic from an opposing approach as shown in Fig 3-1 by dashed lines.

The small range of vehicle speeds and an uninterrupted, short travel path make it desirable to use right-turn travel times to estimate movement flows. Although there is some variability in right-turn times, a range of times can be measured for a corner which includes the travel times of most drivers making a right turn. In turn, knowledge of the range of likely right-turn times makes it possible to estimate the number of right turns that actually occur at a corner. This method of estimating right turn flows is developed in detail in this chapter.

RIGHT-TURN TIMES

Distance and speed alone determine travel time for a vehicle traveling at constant speed. At a given corner, the right-turn path length is virtually constant. In addition, the speed at which a driver can safely maneuver his vehicle through a right turn without encroaching on other lanes or mounting the curb is limited by the curb return radius at the corner. For a curb radius of 20 ft, a typical driver can safely turn right at about 7 mph (side friction factor = 0.17). The time required for a vehicle to travel 30 ft is shown in Table 3-1 for several constant speeds.

Since most vehicles at a corner do not travel at constant speeds, there is some variability in right-turn times. In particular, the first vehicle in a queue at the intersection must accelerate from a stop as it makes a right turn. Other queued vehicles, as well as free flow vehicles, complete the movement at much more nearly constant

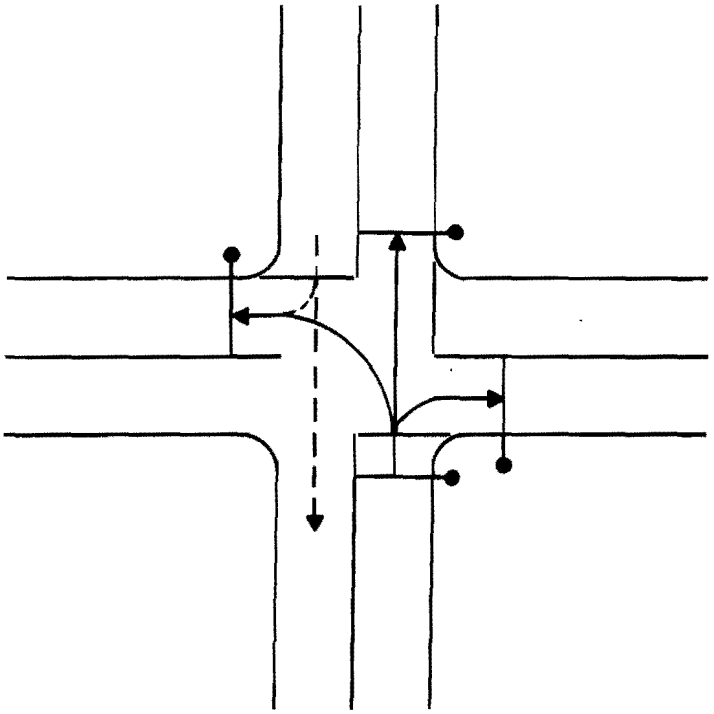


Figure 3-1. Paths for right-turn, left-turn, and straight movement.

TABLE 3-1. TIME REQUIRED TO TRAVEL 30 FT AT CONSTANT SPEED

Speed (mph)	5	7	10
Time (seconds)	4.1	2.9	2.0

speeds. As a result, the right-turn times of the first queued vehicles are significantly longe than those of other vehicles. Different average speeds, as well as small accelerations and decelerations, cause some additional variability in right-turn times.

Intersections with more than one outbound lane have two or more right-turn paths. Figure 3-2 shows an intersection with two outbound lanes. Although Path A may be a prohibited movement, a significant number of drivers will select it; therefore, it must be analyzed. Path A has a longer turn radius, resulting in higher speeds, but it is also a longer travel path. Therefore, the right-turn time by Path A may or may not be longer than by Path B.

For example, at an intersection with a 20-ft curb radius, if $R_A - R_B = 10$ ft, then the effective curb radius for Path A is 30 ft. A driver on Path A could safely increase his speed to about 8.7 mph, whereas a driver on Path B would have to travel at about 7 mph. The length of Path A would be about 47 ft, and the length of Path B between the same detectors would be only 41 ft. The time for a vehicle on Path A to travel 47 ft at 8.7 mph is 3.7 seconds, while the trip on Path B would take 4.0 seconds. This example is summarized in Table 3-2. It is interesting that the additional outbound lane had little effect on the right-turn time.

TABLE 3-2. RIGHT-TURN TIMES FOR ALTERNATIVE PATHS

Path	Effective Curb Radius (feet)	Allowable Speed (mph)	Length of Path (feet)	Right-Turn Time (sec)
A	30	8.7	47.1	3.7
B	20	7.1	41.4	4.0

In addition to dispersion of right-turn times at a given corner, turn times vary from corner to corner. Such variance is almost entirely explained by intersection geometry. As mentioned previously, different curb

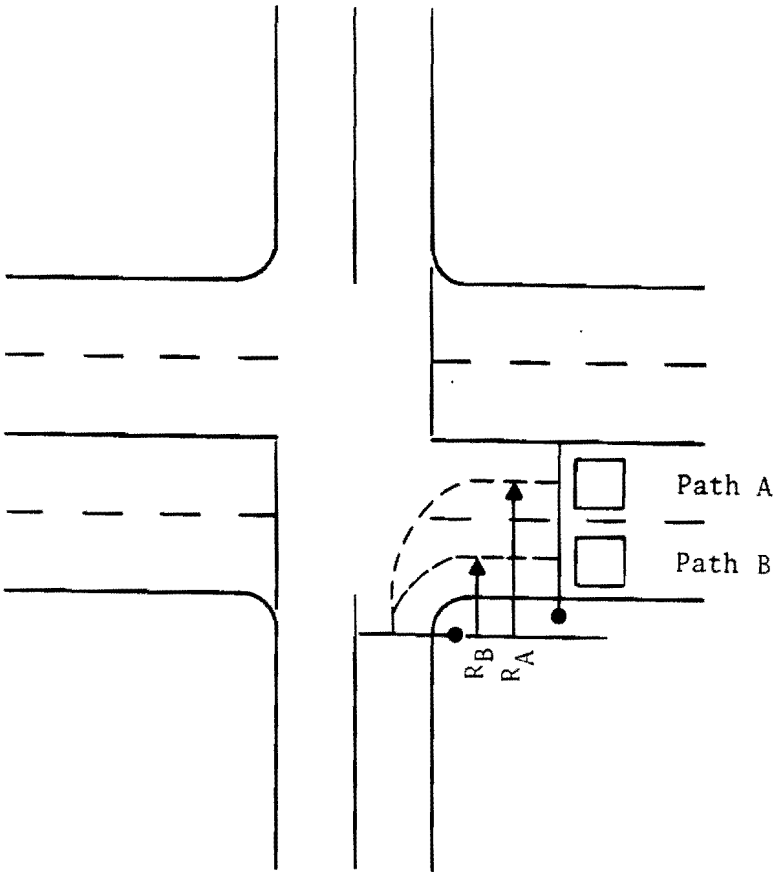


Figure 3-2. Paths for right-turn movements for 4X2 intersection.

radii result in different maximum safe speeds. Also, detector placement, which determines the length of the traveled path, varies across sites.

DETECTOR PLACEMENT

Although detectors should be placed as close to the intersection as possible on all approaches to create the shortest path and reduce the effects of acceleration and deceleration, detector placement often depends on other factors. Wheel detectors should be oriented across the lane in such a way that all wheels on one axle are detected simultaneously. On multiple outbound lanes this may require that detectors be placed much further from the intersection than on single lane approaches. The detector placement is also influenced by driveways and exits or entrances along the approaches. All of these factors cause variability in right-turn times across sites.

FORECASTING RIGHT TURNS

The proposed solution to the problem of obtaining automatic turning movement counts involves estimating the right-turn flows from average right-turn times and known detection times. If all vehicles at a corner complete the right turn movement in the same time, ΔT, then the time the vehicle crosses the Departing Detector B is always

$$T_B = T_A + \Delta T$$

where T_A is the time the vehicle crossed Inbound Detector A. The detector placements are shown in Fig 3-3.

It was shown in the previous section, however, that right-turn times are not constant. A range of possible right-turn times exists, so that the exact time of arrival at the departing detector for a vehicle cannot be predicted. Instead, the earliest arrival time can be estimated as

$$T_{BE} = T_A + \overline{\Delta T} - S_1$$

and the latest arrival time can be estimated as

$$T_{BL} = T_A + \overline{\Delta T} + S_2$$

where $\overline{\Delta T}$ is the average right-turn time at the corner, and S_1 and S_2 are the maximum deviations from the average time. The average and the deviation are for all vehicles except those that are stopped at the intersection and first in the queue before they turn.

Since the first driver in the queue usually takes longer to turn, an additional term must be added to each equation:

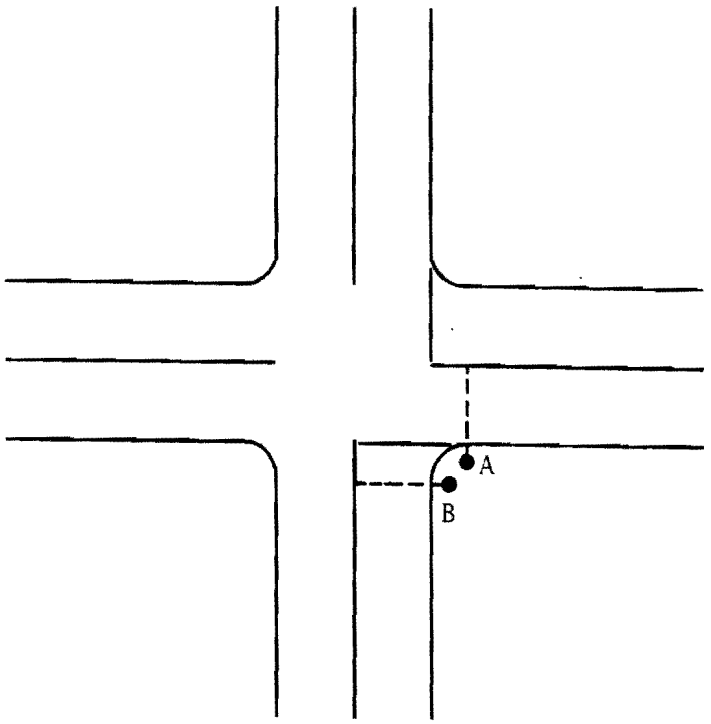


Figure 3-3. Schematic detector placement for forecasting right turns.

$$T_{BE} = T_A + \overline{\Delta T} - S_1 + \alpha Q \quad (3\text{-}1)$$

$$T_{BL} = T_A + \overline{\Delta T} + S_2 + \alpha Q \quad (3\text{-}2)$$

where Q = 1 if the vehicle is first in the queue, and Q = 0 otherwise. The above equations can be used to forecast the earliest and latest possible arrival times at Detector B for a right-turning vehicle crossing Detector A at time T_A. Before the equations can be used, though, ΔT, S_1, S_2, and α must be determined for each corner.

Inbound Detection Time

Recording the time a vehicle crosses Inbound Detector, T_A is the first step in the forecasting procedure. It is important that this time be measured the same way for all vehicles, such as the time the front of the vehicle, or the rear axle crosses the detector. There are several good ways to record the time, and the best way is often determined by the detector type. For the easily installed pneumatic pressure tube, both axle detection times can be recorded, but it is necessary to use only one T_A per vehicle. If it can be shown that two or more consecutive detections were caused by a single vehicle, then one of them can be used as the T_A. Sometimes this is easily seen. For example, if two detections occur 0.70 sec apart, then a vehicle with 11 ft axle spacing traveling 10.7 mph could have crossed the detector. It is also possible that two consecutive vehicles with 17 ft between them (25 ft between axles, minus 8 ft in overhangs), both traveling at 24 mph, crossed the detector, but it is unlikely that vehicles traveling at that speed would be so close. Vehicles traveling further apart would have to be going even faster, so it can probably be assumed that one vehicle caused both detections.

It is not always so easy to determine whether a single vehicle caused two axle detections. A single vehicle traveling 6 mph while preparing to turn, will cause two detections about 1.2 seconds apart; whereas, two vehicles traveling 20 mph through the intersection about 35 ft apart will also cause detections 1.2 seconds apart. In order to estimate the range of uncertainty at a typical urban intersection, 86 right-turning and straight through vehicles were observed at the intersection shown in Figure 3-4. The speed limit on Woodrow was 35 mph, and 40 mph on Koenig. The results are summarized in Table 3-3. The results of this study indicate that the range of uncertainty is small, and that a value of 1.1 second can be used to distinguish between the axles of a single vehicle and those of consecutive vehicles at this intersection. The results of this study are not necessarily valid for other intersections.

Mean Right-Turn Time

The mean right-turn time can be estimated from a random sample of drivers at the corner. To be practically certain that the relative difference between the estimated mean from the sample and the true mean will be no greater than D, the sample must be at least of size n, such that

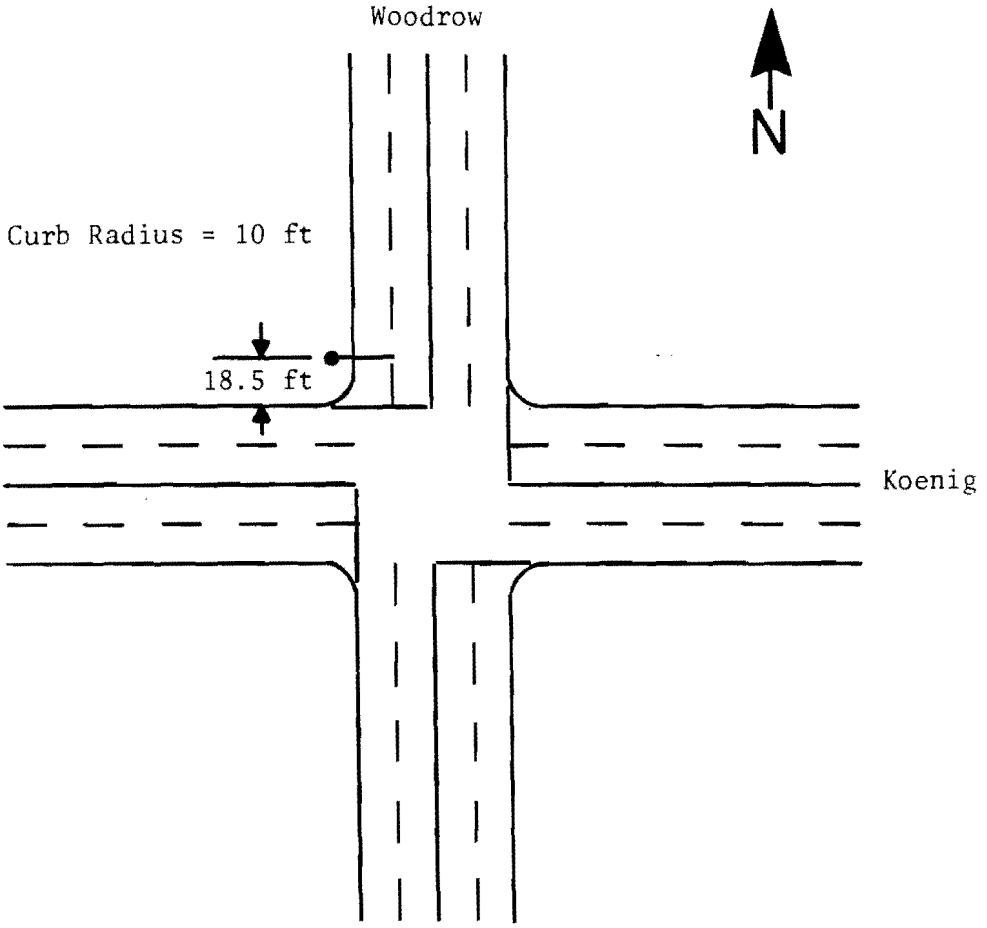


Figure 3-4. Plan view of Koenig Lane and Woodrow Avenue intersection, Austin, Texas.

TABLE 3-3. OBSERVATIONS AT KOENIG AND WOODROW INTERSECTION

	Time Between Axle Detections (seconds)		
	$\Delta t < 1.0$	$1.0 \leq \Delta t \leq 1.2$	$1.2 < \Delta t < 4.5$*
Number of Individual Vehicles With	83	3	0
Number of Consecutive Vehicles With	0	3	53

*Times between axle detections greater than 4.5 seconds were not recorded.

TABLE 3-4. REQUIRED SAMPLE SIZES FOR ALLOWABLE ERRORS

D(%)	5	10	15
n	52	13	6

$$n = (k^2V^2)/D^2 \tag{3-3}$$

where V is an estimate of the coefficient of variation. V can also be written as

$$V = S/\bar{X} \tag{3-4}$$

where S is the standard deviation, and $\bar{X}$ is the mean. The value of k determines the probability that the sample result will have a relation error no greater than $\pm D$. Setting k = 3, Equation 3-3 becomes

$$n = (9V^2)/D^2 \tag{3-5}$$

Data was collected at a typical urban intersection as shown in Fig 3-5 to estimate the coefficient of variation. Sixty right-turning vehicles were observed. For the 51 that were not the first in the queue, the mean right-turn time was 3.92 seconds, and the standard deviation was 0.486 seconds. From Equation 3-3,

$$V = 0.486/3.92 = 0.124$$

For D = 10 percent,

$$n = 9(0.12)^2/(0.10)^2 = 13$$

Required sample sizes for different allowable errors are summarized in Table 3-4. Assuming V = 0.12 is approximately correct for all intersections, a sample of 13 to 15 right-turning vehicles can be used to estimate the average right-turn time at a corner. The estimated mean may be off by as much as 0.40 seconds at a typical corner, thus, larger samples are much preferred.

Stopped Vehicle Factor

As mentioned previously, the first stopped vehicle at an intersection takes significantly longer than other queued and free flow vehicles to complete a right turn. Because of this difference, a factor, α is included in the forecasting equations to account for the additional time required by the first stopped vehicle. Data was collected at three Austin, Texas intersection corners to estimate α. The results of the study are summarized in Table 3-5. The estimates of α varied from intersection to intersection, probably because of intersection geometry and detector placement. Although the sample sizes were small, and consequently, ΔT, for Intersections 2 and 3 could have errors of more than ten percent, these results suggest that α should be estimated separately for each intersection corner. Estimating α could take quite a bit of time, since only one vehicle per corner can be

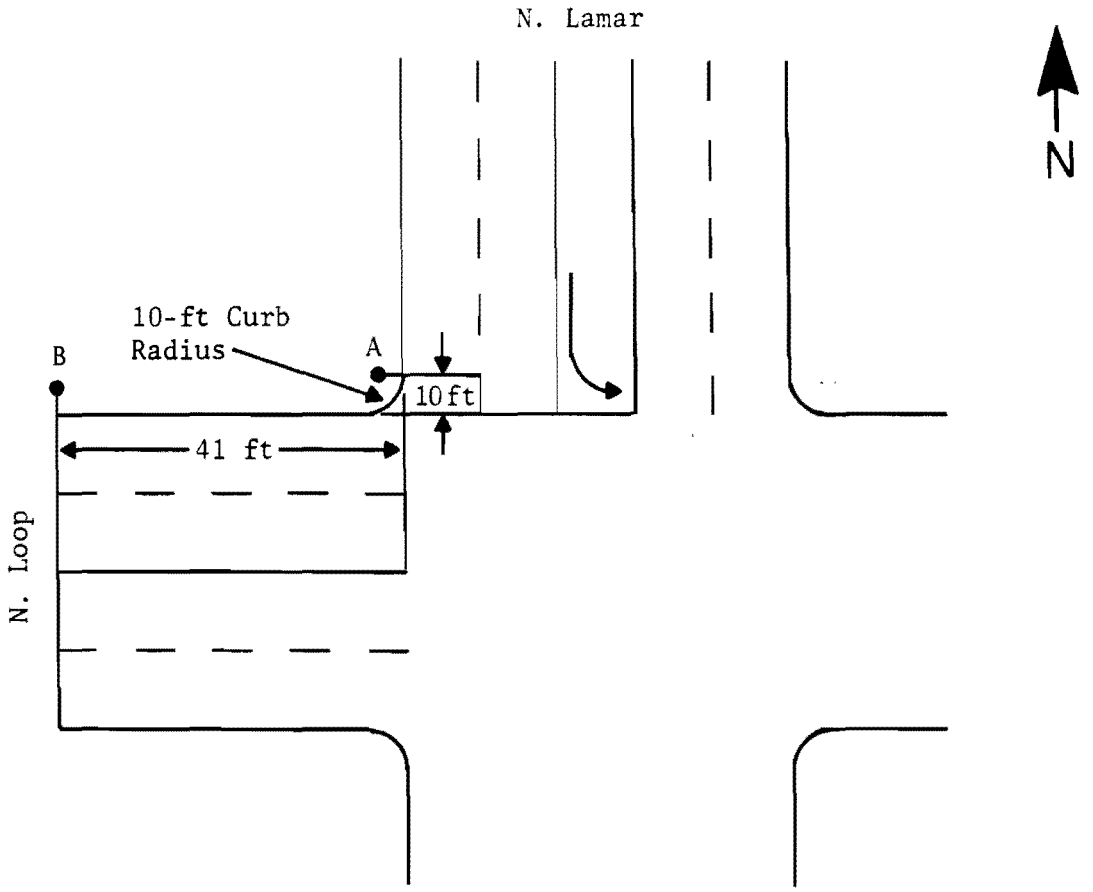


Figure 3-5. Plan view of N. Lamar Blvd. and North Loop intersection, Austin, Texas.

TABLE 3-5. AVERAGE RIGHT-TURN TIMES FOR VEHICLES FIRST IN THE QUEUE AND NOT IN THE QUEUE AT THREE INTERSECTIONS IN AUSTIN, TEXAS

	Intersection	First-in-Queue		Not First-in-Queue		Stopped Vehicle Factor
		Vehicles Observed	$\overline{\Delta T}_1$	Vehicles Observed	$\overline{\Delta T}_2$	$\alpha = \overline{\Delta T}_1 - \overline{\Delta T}_2$
1	45th & Bullcreek (SE Corner)	35	4.53	85	4.18	0.35
2	N. Lamar & North Loop (SW Corner)	13	5.42	88	4.73	0.69
3	Koenig & Woodrow (NW Corner)	11	5.94	34	5.00	0.94

timed during each phase to determine the right-turn time for the first stopped vehicle, and it may not necessarily turn right. Future data analysis may lead to tabularized values of α to aid the user in different situations.

Deviations From the Mean

Earlier in this chapter equations were developed for forecasting the earliest and latest possible arrival times at an outbound detector or a right-turning vehicle. These times are used to set the bounds for the forecasted right-turn interval, and any detection occurring at the outbound Detector B such that $T_{BE} < T_B < T_{BL}$ can be assumed to have been caused by a right-turning vehicle. At a corner, the interval width is determined by the maximum deviations from the mean right turn time, S_1 and S_2. Figures 3-6a and 3-6b show the frequency of observed right turn times at two typical urban intersections. The deviation in the lower direction, S_1, is equal to 4.7 - 3.5 = 1.2 seconds for the corner in Fig 3-6a. At that same corner, S_2 = 6.1 - 4.7 = 1.4 seconds. In this case, S_1 is approximately equal to S_2, and the data appears to be normally distributed. For the case shown in Fig 3-6b, S_1 = 0.90 seconds and S_2 = 1.80 seconds. Slow turning vehicles cause the longer tail at the upper end of the data, but the lower end on this plot, as well as for most other corners, has no tail because maximum vehicle speeds are limited by the radius of the turn.

If the interval given by $S_1 + S_2$ is sufficiently long, vehicles completing other movements may cross Detector B within the interval and be miscounted as right turns. For example, as shown in Fig 3-7a, vehicle X may cross Detector A at time T_A,while vehicle Y on the opposing approach waits to turn left. If vehicle X travels rapidly straight through the intersection and has a sufficient gap behind him, then vehicle Y can turn left immediately behind him and cross Detector B at time T_B such that $T_{BE} < T_B < T_{BL}$. This is shown in Fig 3-7b. It is also possible for an opposing vehicle to turn left in front of a vehicle crossing Detector A, and arrive at B such that $T_{BE} < T_B < T_{BL}$, but this is not as likely to occur. Thus, the interval determined by the minimum and maximum right-turn times may be too long, and a significant number of vehicles completing other movements may be miscounted as right turns.

At three intersections studied, the range of right-turn times was approximately 2.75 seconds. This interval was found to be sufficiently long to permit other movements to be counted as right turns. Since the tails of the right-turn time distribution are thin and short, one may be able to reduce the interval somewhat, without losing too many correct right-turn counts, yet eliminating many of the miscounts. For the case of the data in Fig 3-6a, if the right-turn times are fitted to a normal distribution with S = 0.482, 95 percent of the right-turn times will occur between 3.8 and 5.7 seconds. This reduces the interval to 1.9 seconds, and the resulting S_1 and S_2 values would be the same and equal to 1/2 (1.9) = 0.95 seconds. By reducing the interval almost 0.40 seconds at each end, much of the miscounting is eliminated. It is not known at this time what the total right-turn counting error of the device would be, but S_1 and S_2 values of 0.80 to 1.00 seconds are recommended for testing. Testing results may provide user guidelines, and more information regarding accuracy.

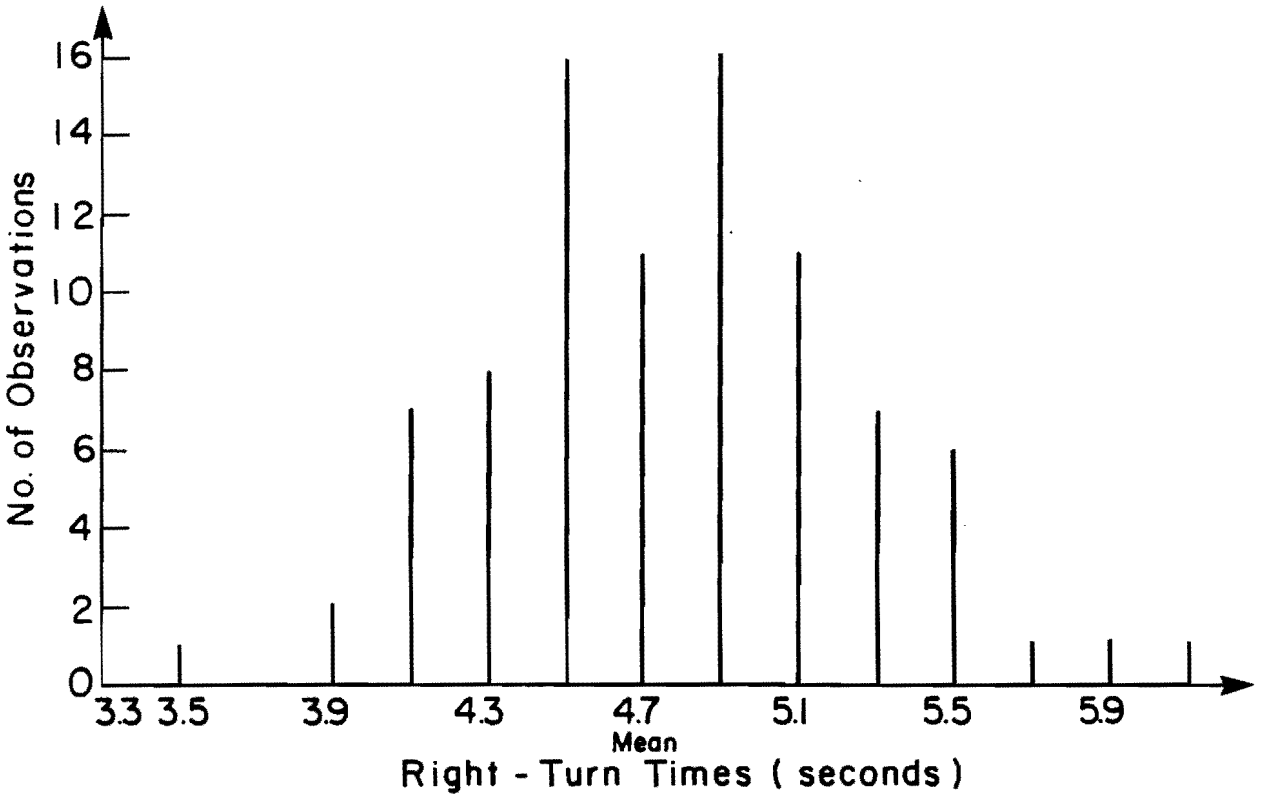


Figure 3-6a. Right-turn times for vehicles other than the first in the queue, collected at the intersection of N. Lamar and North Loop, Austin, Texas (88 observations).

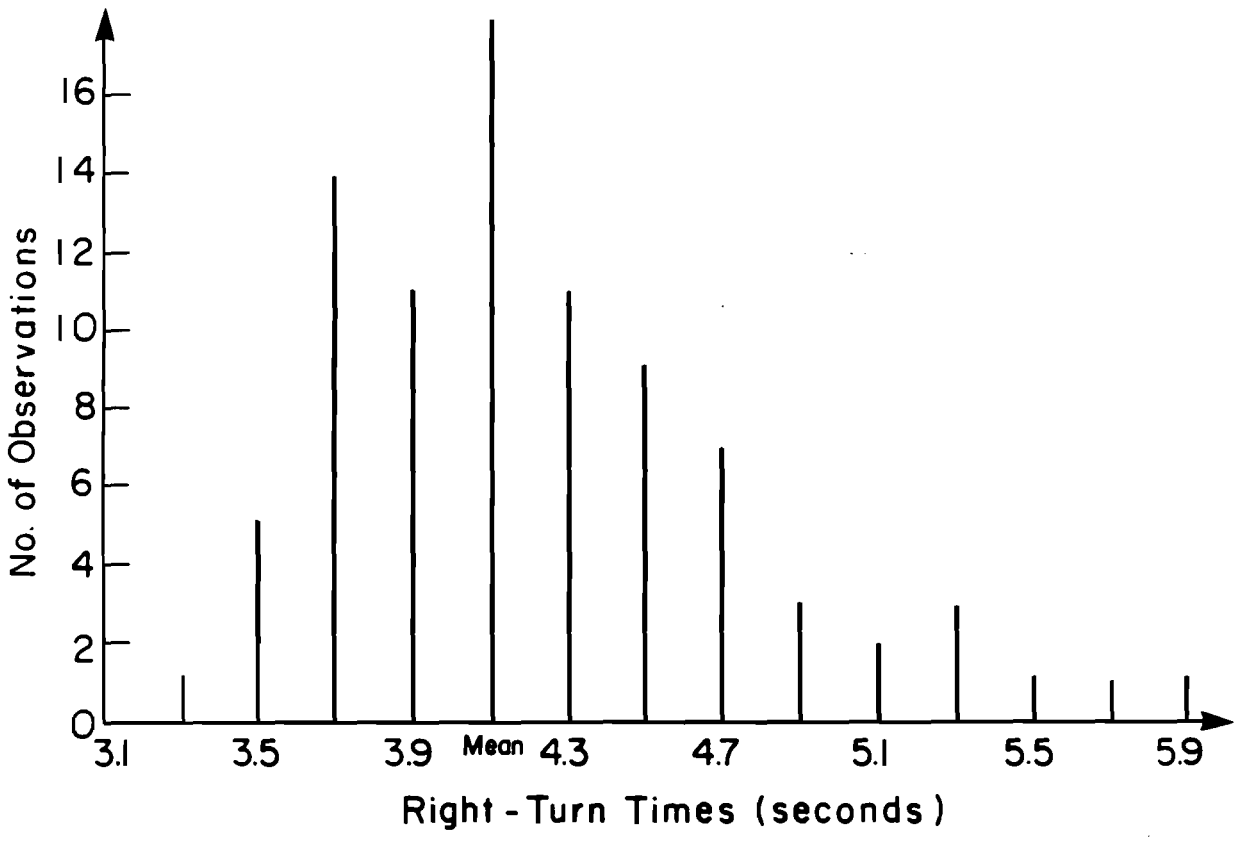
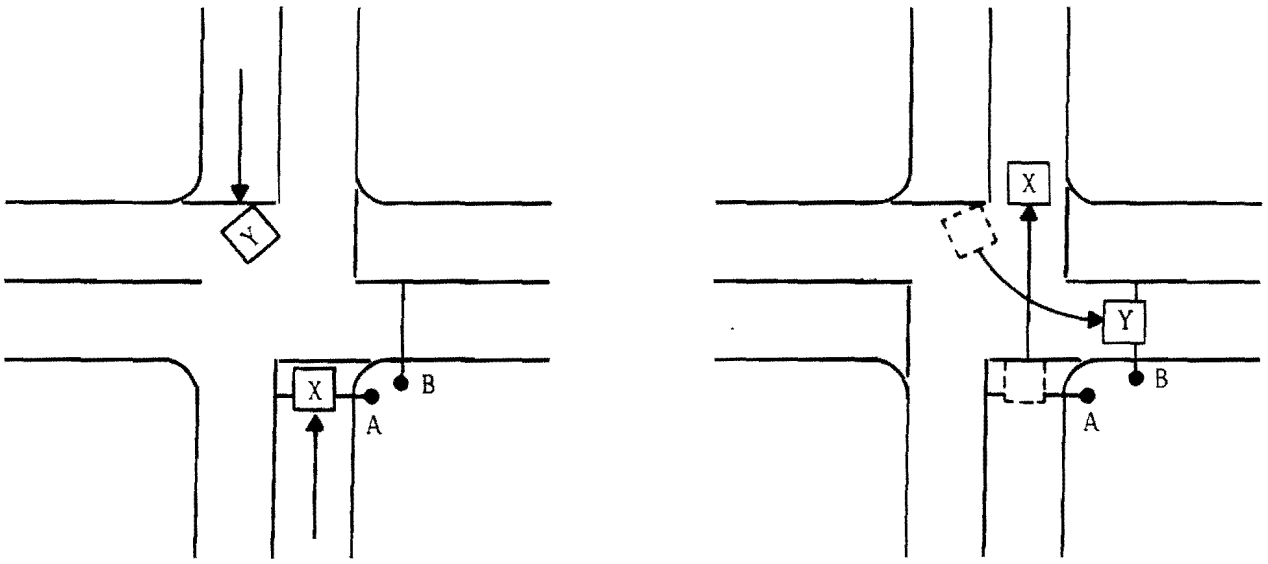


Figure 3-6b. Right-turn times for vehicles other than the first in the queue, collected at the intersection of 45th St. and Bullcreek, Austin, Texas (96 observations).



(A) (B)

Figure 3-7. Schematic representation of potential right-turn miscounting.

Summary

It is important to note at this point that determining the best values of the parameters to be used in the forecasting procedure is not an easy task. A tremendous amount of time and patience is required to collect the data that is needed for the analysis. The purpose of the discussion presented here is to show that good estimates of $\overline{\Delta T}$, S_1, S_2, and α can probably be obtained by the typical user. Testing of an already designed version of the automatic turning movement counter at various intersections will provide additional information to aid the user.

METHOD OF ESTIMATION AND ITS ACCURACY

Once all of the parameters in the forecasting equations have been determined, the equations can be used with detection times to estimate the number of right turns at a corner. The user should be able to adjust $\overline{\Delta T}$, α, S_1, and S_2 for each corner. When the inbound detection time, T_A is measured (preferably at the rear of the vehicle or rear axle, so the right turn time is measured the shortest possible way) and is recored, Equations 3-1 and 3-2 can be applied to generate the earliest and latest likely arrival times at Detector B for a right-turning vehicle. The earliest arrival time is generally more than three seconds after T_A, so the computer has sufficient time to generate both T_{BE} and T_{BL} before the time T_{BE}. Then any detection at B that occurs during the interval bounded by T_{BE} and T_{BL} can be interpreted as a right turn and should be recorded as such.

Because the right turns are only estimated, not actually observed, results from this method will have some error associated with them. In addition, there is error associated with the estimates of $\overline{\Delta T}$, α, S_1, and S_2. It is difficult to predict what the accuracy of the right turn counts will be for an actual system. It is doubtful whether 95 percent or even 90 percent accuracy in the right turn counts can be obtained. It is very promising, however, that up to 80 or 85 percent accurate estimates will result, and such accuracy should be sufficient for many applications of traffic data.

CHAPTER 4. SYSTEM DESIGN

A number of alternative hardware concepts were evaluated for implementation of the counting algorithm. The hardware system was considered to be composed of three basic components. These included (1) a logic unit and recording system which would apply the algorithm to detector inputs and record field counts and/or detection times, (2) an interface device which would connect detector outputs to the logic unit, and (3) a detection system. Considerations for the design or selection of each of the components included reliability, cost, adaptability, and availability.

DETECTION SYSTEMS

The basic counting methodology was designed to be largely independent of the type of detector utilized. The use of axle-counting versus vehicle-counting detectors would require either a hardware or a software compensation mechanism for the number of axles per vehicle as vehicle, not axle, counts are usually the usually desired statistic. However, due to the availability and long-time use of various types of axle counting devices, early experimentation centered around axle-counting equipment.

Pnuematic road tubes and a piezoceramic axle-counting device were tested extensively. The principal advantages of road tubes are low cost and availability. Time and effort required for installing of eight road tubes in the vicinity of a busy intersection represent a significant disadvantage, however. Reliability in terms of keeping a road tube in its installed position, especially when subjected to high-volume, sometimes high-speed traffic, is also a problem.

An axle detector using piezoceramic crystals, and developed under CTR Research Study Number 3-8-81-310, was also examined extensively. The piezoceramic material utilized in these devices has the characteristic of emitting a voltage when deformed by the tires on the axle of a passing vehicle. The specially fabricated axle-detection devices which were tested, consisted of a string of piezo-crystals electrically connected in series and spaced closely enough to preclude the passage of a vehicle tire without deforming of at least one crystal. The crystals and connecting wires were encased in a molded rubber envelope which sealed the system from moisture damage and provided a convenient strip for mounting on the pavement surface. This design allows the manufacture of detectors in any desired length and makes lanewise axle detection possible. Installation involves less effort than conventional road tubes. The installation process merely requires application of a pressure-sensitive adhesive to the bottom of the molded rubber detector and placing the unit on the pavement surface at the chosen location. A strip of ordinary asphalt roofing material laid over the strip can provide additional resistance to displacement of the detector under traffic. Cost of a mass-produced piezo type detector would be approximately

the same as by conventional road tubes. Service life for intersection counting applications would be approximately equivalent to road tubes. Although the piezeceramic type detectors offer a solution to several of the principal problems associated with road tubes, they detect axles, not vehicles, and are not ideally suited for automated turning movement counting.

Therefore, parallel to the effort examining axle detectors, an examination of potential vehicle counting devices was also conducted. A variety of detection systems was examined, however the most promising was an infra-red light beam system. This system consists of a small battery powered combination transmitte-receiver unit which emits an infra-red light beam across the detection area and a reflector which returns the beam to the adjacent receiver. Any object that passes through the detection area blocks the light beam and causes a switch closure in the connected electronic circuit. The switch remains closed as long as the beam is interrupted. Due to rather high sensitivity of the transmitter-receiver, a wide variety of reflection devices can be successfully used. Raised reflective traffic buttons and reflective sheeting were found to be quite adequate for distances of interest.

Field tests of the infra-red detector, which is manufactured by OPCON were conducted to quantify the distances over which the system would operate reliably. The tests indicated that reliability is a function of the size and sensitivity of the reflector. Reflective traffic buttons mounted on the pavement surface or on the top of a curb were found to be completely reliable for distances up to 40 feet, while the range could be extended to 100 feet or more by enlarging the size of the reflector surface. The most practical reflector system for distances over 40 feet was found to be ordinary reflective sheeting placed on a vertical curb face.

With the reflector mounted on the pavement surface and the transmitter-receiver unit a few inches above the surface, the infra-red detector will react to the passage of axles, while merely elevating the transmitter-receiver and/or the reflector will effectively cause the system to react to vehicle units. While elevation of the reflector may not be practical in many situations, elevation of the transmitter-receiver, which weighs less than one pound and measures approximately six, by two by two inches, can be very easily accomplished. Vehicle detection can also be achieved by mounting the transmitter-receiver unit directly above a reflector mounted flat on the pavement surface.

Because of the ease with which it can be installed, and the fact that it enables the direct counting of vehicles, rather than axles, the infra-red detection system is considered to be the most appropriate for automated turning-movement counting over short periods of time. This system can also be used for long-term installations with appropriate mounting of the transmitter-receiver units.

Conventional inductance loop detectors can be used for long-term installations where the expense of installation is considered to be justified. Inductance loop detectors would provide essentially the same type of vehicle detection output as the infra-red detectors described in preceeding paragraphs. Due to the cost and effort required for installation, however, installation of loops is only recommended for long-term counting operations. Locations for loop detectors, like other detector systems must be chosen carefully so that reliability of the

counting operation can be maintained. The reliability discussions outlined in Chapter 3 must be considered. The most desirable means of choosing permanent detector locations would be a trial and error process in which surface mounted loops are taped on the road surface prior to selecting a permanent location.

All the detector systems evaluated require installation of wires for connecting each detector to the logic unit. A simple remote tellemetry system is needed to solve the inherent problems that are associated with hardwire connection of detectors and logic units.

LOGIC UNIT AND RECORDING SYSTEM

Early in the design process, the decision was made to utilize off-the-shelf micro-computer hardware due to its minimal cost, reliability, and availability. A micro-computer provides flexibility in recording traffic counts in solid-state memory or in some form of magnetic media such as "floppy disks". It also provides a convenient means for modifying software-based counting instructions. Additionally, when not in use for turning movement counting, the device can be used for a variety of other data-processing purposes.

INTERFACE DEVICES

Considerable effort was expended in evaluating the relative merits of designing or select an interface device. An interface device is needed to receive information (in the form of switch closures or voltage changes) from as many as eight detectors plus a signal controller and transfer this information to the microcomputer in an acceptable format in real time.

Two possible courses of action became apparent, and both were thoroughly examined. A custom-made interface device could be designed and built for use with one or more types of closed-architecture micro-computers. The complexity and resulting cost of such a device would, among other things, be determined by the type or types of micro-computers to which it would interface. The alternative course of action would utilize an open-architecture type of micro-computer and an off-the-shelf (plug-in) analog to digital (A-D) board as an interface. The costs of open-architecture machines, such as the IBM PC and its family of clones, are generally higher than the closed-architecture machines such as the Epson and other "lap top" models. Costs of custom made interface devices are, however, significantly higher than off-the-shelf A-D boards. Thus, when the investigation began, the two courses of action seemed to involve approximately equal overall costs for hardware.

Initial discussions with SDHPT contact individuals for the study indicated that a custom-made interface device and a closed arcitecture "lap top" type of microcomputer was the preferable choice. A preliminary design of an interface device for an Epson computer was developed along with estimates of production costs. While the design of this interface device progressed, information regarding new developments in low-cost A-D data processing was collected. As the design of the custom-made interface device for the Epson neared completion, several events tended to change the direction of the interface development effort.

Continuing dialog with SDHPT contact individuals indicated that the Epson computer might very likely not be the machine of choice. This meant that certain elements of the interface design would require modification. Additionally, cost estimates for designing and building the interface system were higher than expected. At this time, it was determined that the combined cost of a closed-architecture micro-computer and the associated interface system would exceed the cost of an open-architecture machine and an off-the-shelf A-D board. The price differential was further impacted by the entry into the market place of a significant number of manufacturers offering A-D hardware for very low-priced open-architecture "look alike" micro-computers. Therefore, a decision was made to abandon the custom-designed interface system development effort. The alternative of an open architecture micro-computer with an off-the-shelf A-D circuit board was selected.

Subsequent to this decision, a grant to the faculty researchers from IBM through the "Quest" Program at The University of Texas at Austin made a portable IBM PC and a suitable A-D circuit board available for use in this study. This system was utilized in all further testing of the turning-movement counting algorithm.

TECHNICAL REQUIREMENTS

The choice of generic off-the-shelf hardware for the logic unit and interface device dictated that virtually all aspects of the turning-movement counting metholology should be implemented in software rather than hardware. In addition to being the lowest cost method, this implementation scheme provided the significant advantage that changes and updates to the methodology could be easily accomplished throughout the development process. For practical use in a turning-movement counter system, the hardware components are required to meet several minimum specifications.

A minimum of nine data-input channels are required to sense the status of eight detectors and the traffic signal controller continually. Since the occurrence of switch closures, rather than voltage levels, can be easily provided by the detectors and an ordinary relay can provide the same type of "state" information about the signal controller, only digital, instead of analog, input channels are required. The use of digital (TTL) logic represented a significant cost savings for the interface device because most available A-D circuit boards for 16-bit microprocessor-powered computers (such as those using the Intel 8088 chip) provide 16 digital,TTL-compatible, input channels. The typical number of analog channels varys with manufacturers, but the number of channels that are provided on a single circuit board is generally less than the required nine. Thus, the use of digital inputs means that only one circuit board is required for making turning-movement counts at even the most complex signalized intersections. A basic system schematic is provided as Figure 4-1.

The logic unit must have sufficient operating speed to permit checking the status of all nine digital input channels, performing necessary calculations, and storing results with a frequency of about 10 repetitions per second. This operating speed is a function of the basic speed of the microprocessor as well as the efficiency of the instruction code which directs its operation. FORTRAN 77 (ANSI Standard 3.2) was chosen as the computer language in which the code would be written. FORTRAN represented a practical compromise between

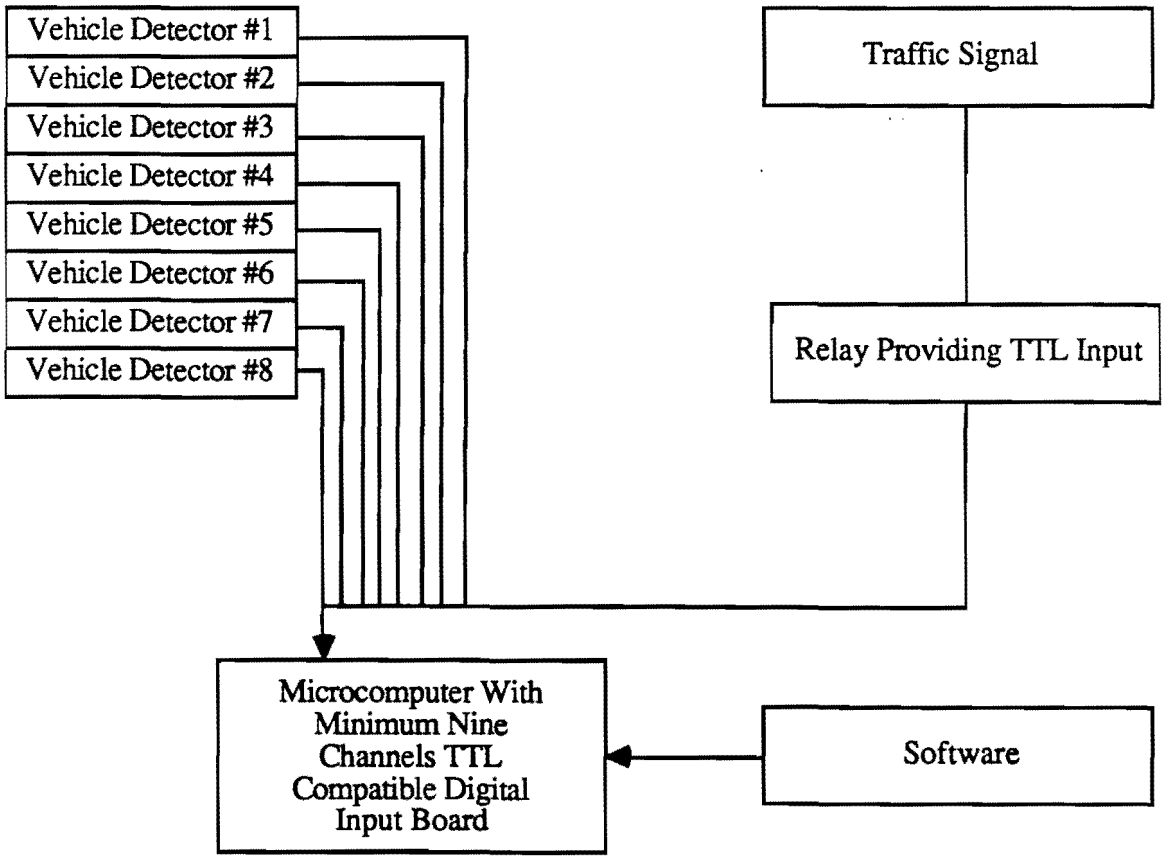


Figure 4-1. Basic system schematic.

faster, lower-level languages, which require much more code development effort, and slower, higher-level languages which might offer less code-development effort. ANSI Standard FORTRAN provides the added benefit of being highly transportable, since the code can be compiled and run without modification on virtually any machine having an appropriate compiler. A listing of the program for the worst case of an unchannelized 4-leg intersection is provided as Appendix A.

The portable IBM PC upon which the technique was tested was powered by an Intel 8088 microprocessor which has a clock cycle of 4.77 megahertz. Bench tests were conducted regarding the speed with which a FORTRAN routine could check the status of nine digital input channels on an IBM Data Acquisition and Control Adapter (DAC) board. The test code did no calculations but simply cycled through the digital input status checks as rapidly as possible. Results of the bench test indicated that the code could repeat the checks of the nine input channels approximately 1500 times per second, which was at least, two orders of magnitude faster that the minimum requirement.

Real time forecasting of the durations of right turn movements as well as timewise recording of the numbers of turn movements implied that the logic unit must be equipped with a real time clock. The minimum clock resolution would need to be 0.1 second, since as explained earlier, tenth second time increments could be significant to the right turn forecasting process. The IBM PC utilized in testing had a built-in clock with a resolution of approximately 0.05 seconds. This clock resolution proved to be quite adequate.

COUNTING SYSTEM OPERATION

The operation of the algorithm with two phase signal control on a multilane, unchannelized intersection, as depicted in Figure 4-2, is explained as follows. As indicated in the figure, phase "A" represents a green phase for the north-south street. During this signal state, detectors numbered 6 and 8 can be accessed by straight manuevers on the north-south street and right-turns on red (RTOR) from the east-west street. Since the RTOR manuevers are counted directly by detectors 2 and 4, totals for these detectors can be subtracted from totals for detectors 6 and 8 to yield straight movement counts as signal phase "A" concludes. Additionally, detectors 5 and 7 can be accessed by left and right-turning traffic.

As each vehicle crosses detector 1 or 3, a time window is forecast for his arrival at the respective outbound right-turn detector (either 5 or 7). If an arrival occurs at at the appropriate detector (5 or 7) during the window, it is counted as a right-turn from the appropriate approach. Right-turns on green can be subtracted from count totals of detectors 5 and 7 to yield left turn totals for signal phase "A". During signal phase "B", the mirror image of these deductions is utilized.

A potential problem regarding the proper handling of vehicles trapped in the interior of the intersection at the end of a signal phase was discovered during the testing process. To solve this problem, a transition time is provided within the counting software at the end of each signal phase. During this time, traffic which has just received a green signal indication is undergoing its start-up delay and has not yet begun moving. Therefore,

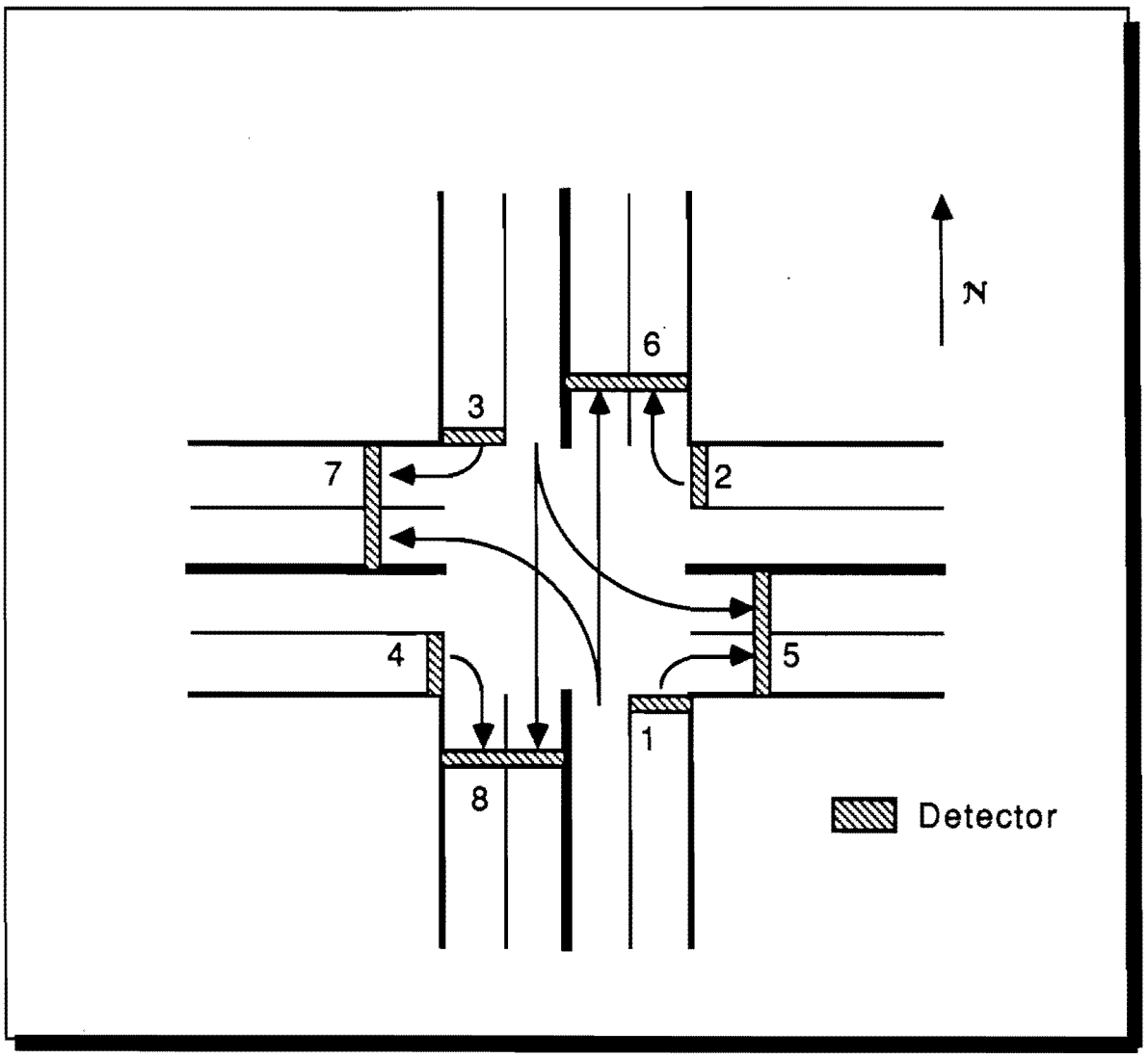


Figure 4-2. Legal traffic operations during phase"A" (north-south green).

during the transition time, all detector relationships of the previous signal phase are maintained allowing any trapped vehicles to be handled properly.

ACCURACY

As noted in the previous paragraph, straight and RTOR movements are counted directly and are therefore, only subject to errors associated with the detection devices. Right-turns on green, however, are deduced from the time window forecasting process. Extensive testing of this process indicated that the only reliable method for estimating right-turning times is through field measurement at each intersection and desirably at each corner of each intersection.

Therefore, a calibration routine was written , which uses the detectors, which have been installed for the counting process, and the microcomputer to estimate turning times. The routine functions as a pre-processor to the counting program and passes, selected window data automatically. Tests of the counting processor, indicate that the accuracy of the right-turn forecasting procedure usually exceeds 90 percent for usual turning movement flow combinations.

CHAPTER 5. RIGHT-TURN TREATMENTS

The studies of right-turn phenomena conducted as part of the turning movement counting effort led to two specific questions regarding the treatment of right-turns at signalized intersections. First, can a rational basis be developed for assessing the relative effects of right-turn and straight-through traffic movements within the context of volume warrants for signal installation? Second, can guidelines for use of right-turn bays and right-turn lanes be developed? These two questions are examined in the following sections.

RIGHT-TURN EQUIVALENCE RELATIONSHIPS

Currently, the Manual on Uniform Traffic Control Devices (MUTCD) does not provide a means for distinguishing between the operational effects of different volumes of turning traffic and straight-through traffic within the volume warrants for signalization given in Section 4c (Texas MUTCD). Turning and straight traffic volumes are normally summed to produce the total approach volumes which are referenced within the warrants. However, it is intuitively obvious that maximum flow rates are not the same for turning and for straight movements. In other words, the time needed for making a turning and a straight movement are not equivalent. Therefore, the evaluation of an approach volume, in terms of its need for signalization, should include some means of accounting for the effects of turning movements. The following is a description of work done toward development of equivalence relationships which could be used in traffic volume warrant analyses for signalization.

Background

A computer file search for information pertinent to the analysis produced several references primarily dealing with signalization concepts. The issue that has stimulated most interest regarding right-turn movements through signalized intersections has been the advisability of allowing and effects of right-turns-on-red (RTOR). McGee, et.al. [Ref 13] developed guidelines for the use of ROTR and evaluated relative effectiveness. The study scope included field and computer simulation data collection, accident analyses, and a signing needs evaluation. The study results support the permissive RTOR rule noting benefits of reduced delay, fuel consumption, and auto emissions, without sacrificing safety. The study also provided evidence, through its collected data, that maximum flow rates are generally smaller for straight as opposed to right-turn movements. The RTOR was presented as a means of increasing maximum right-turn flows and decreasing user delays.

Kraft and Vechten [Ref 9], examined driver responses to the use of various types of right-turn signal indications. The study concentrated upon field collected data at intersections in Washington, D.C. with heavy right-turn demands. The primary measure of effectiveness monitored through the study was the percentage of violations, however, peripherally, analyzed data again confirmed differences in maximum right and straight flow rates. Mamlouk [Ref 11] and May [Ref 12] also studied the costs and benefits of RTOR signalization through studies in Indiana. Both concluded that significant benefits could be derived from selective RTOR implementation.

Development of an Equivalence Relationship

As noted in the previous section, little information regarding equivalence of straight and right-turn movements could be found. However, several references noted differences between right and straight maximum flow rates. Therefore, one logical means of equating the two types of traffic flows might be in terms of their maximum flow rates. A series of experiments were designed to explore the differences in maximum flow rates for a variety of typical geometric and traffic control schemes.

If the equivalence relationships are to be used in warrant studies, the traffic control types which should be studied are those that would most likely be in effect prior to signalization. This consideration led to the inclusion of 2-way and all-way stop as the most likely traffic control features. Street geometry should likely also effect the relative performance of straight and right-turn maneuvers. Therefore, intersections with 4 X 4 and 4 X 2 lane geometries were included. The TEXAS Model for Intersection Traffic [Ref 10], which is a microscopic traffic simulation model was configured with these basic geometric and traffic control features. For each test condition, essentially infinite queues of traffic were generated for the stop-controlled approaches while traffic volumes on the cross street were varied from zero to as much as 2300 vehicles per hour (expressed as the sum of both cross street approach volumes). Specifications utilized in the simulation included 600 feet intersection approach lengths and 20 feet curb return radii.

Results of the 2-way and all-way stop experiments for 4 X 4 and 4 X 2 lane geometries are presented in Tables 5-1 and 5-2. Each of the maximum flow rates presented in the tables represents the arithmetic mean of four replicate observations of 30 minutes of simulated observation time. Graphical presentation of the data for the 4 X 4 lane geometry type under 2-way stop control is shown in Figure 5-1. The figure indicates that the stop-controlled straight movement has a slightly higher flow rate until the total volume of traffic being crossed reaches approximately 800 vehicles per hour. As the uncontrolled crossing traffic volume increases above the 800 vehicles per hour, the maximum right-turn flow rate exceeds the straight flow rate by an increasing margin.

A similar graphical comparison of maximum flow rates for stop-controlled straight and right maneuvers with 4 X 2 lane geometrics is presented in Figure 5-2. In this case the maximum flows are not discernibly different until the "to be crossed" flows exceed approximately 1500 vehicles per hour. This is probably true because the crossing traffic is confined to one lane in each direction. This means that the straight and right-turn

TABLE 5-1. MAXIMUM STOP SIGN FLOW RATES PER LANE FOR 2-WAY STOP CONTROL

Total Uncontrolled Crossing Traffic (Total Both Approaches,vph)	Maximum Per Lane Stop Sign Flow (vph/lane) Intersection Geometry			
	4 X 4	4 X 4	4 X 2	4 X 2
	Movement			
	Straight	Right	Straight	Right
0	380	361	382	392
100	392	364	376	386
300	366	353	348	348
500			327	331
700	333	321	304	302
900			286	285
1100	288	306	262	268
1300			248	247
1500	245	292	210	216
1700			176	196
1900	198	274		
2300	144	253		

TABLE 5-2. MAXIMUM STOP SIGN FLOW RATES PER LANE FOR ALL-WAY STOP CONTROL

Total Crossing Traffic (Total Both Approaches,vph)	Maximum Per Lane Stop Sign Flow (vph/lane) Intersection Geometry			
	4 X 4	4 X 4	4 X 2	4 X 2
	Movement			
	Straight	Right	Straight	Right
100	389	377	377	380
200			369	375
400	365	367	360	366
600			357	360
700			358	360
800	346	347		
1200	340	339		
1350	339	335		

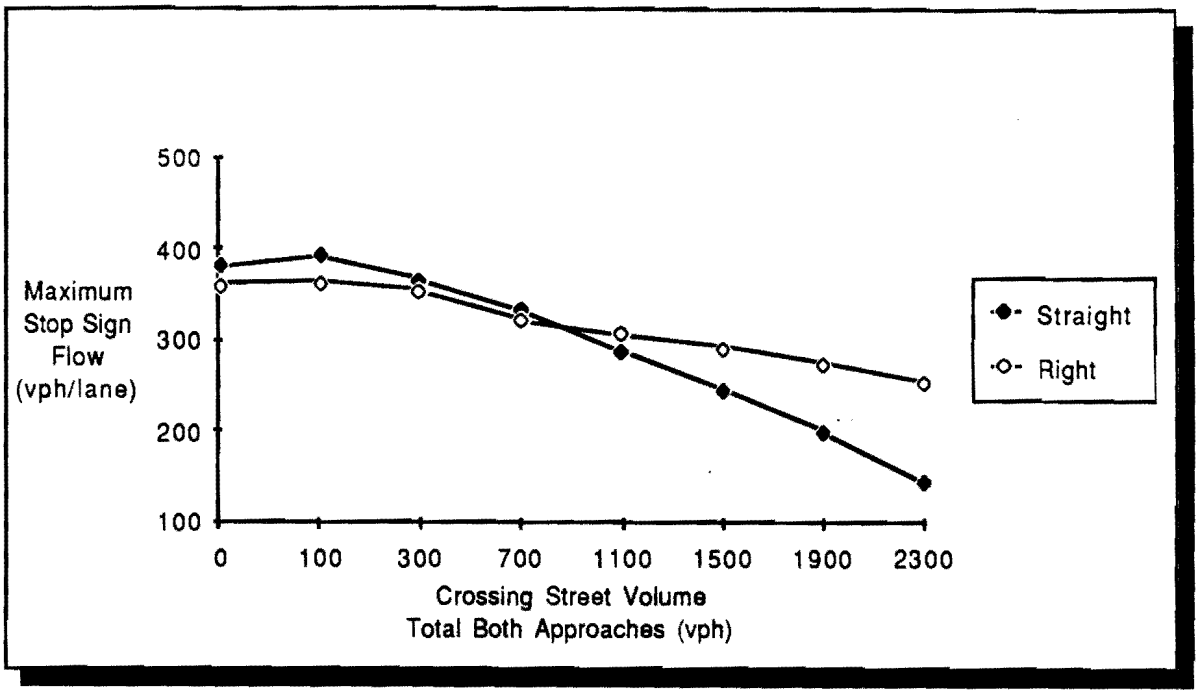


Figure 5-1. Maximum flow rate per lane, 4 X 4 geometry, 2-way stop control.

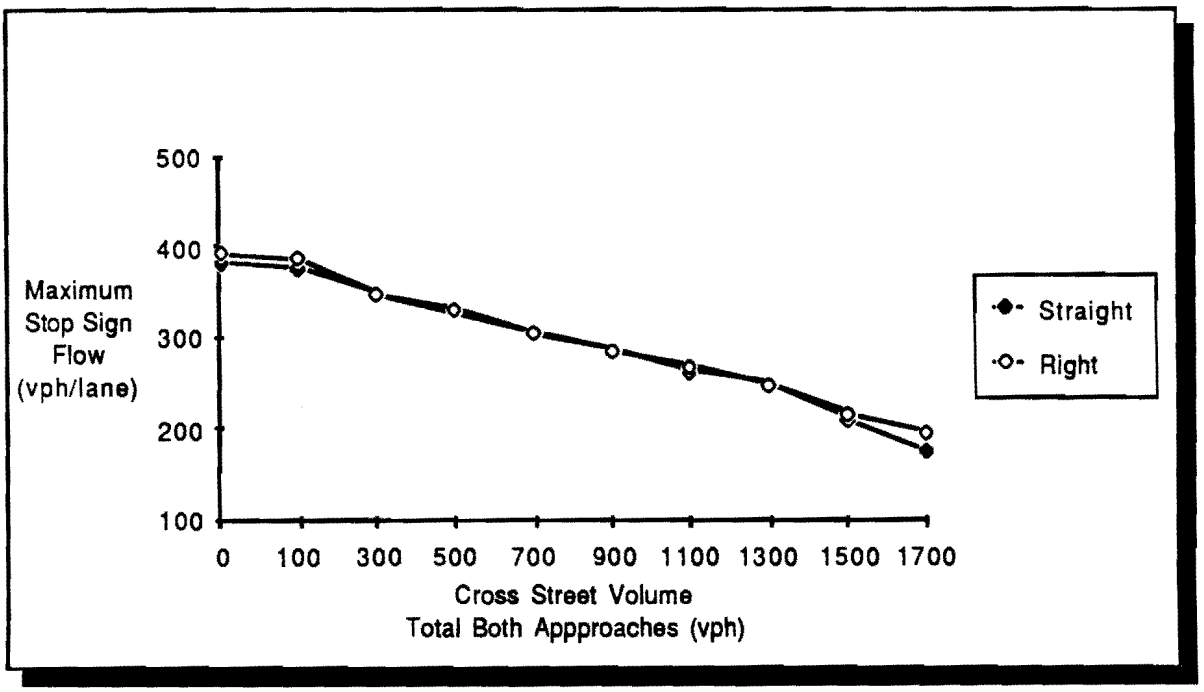


Figure 5-2. Maximum flow rate per lane, 4 X 2 geometry, 2-way stop control.

flows on the four lane street, must utilize the same gaps in the crossing street traffic for executing their maneuvers. The right-turn flows do not have an opportunity to turn into a curb lane which may serve as a "free" right-turn lane in the 4 X 4 case.

The graphical comparisons are further extended to the all-way stop case through Figures 5-3 and 5-4. The trends indicated in the two figures are structurally similar to the cases for 2-way stop control. However, differences in maximum flow rates are generally less significant due to the very orderly sharing of right-of-way which normally occurs at all-way stop intersections.

As noted previously, one means of equating right and straight movements is in terms of their equivalent maximum flow rates. The data of the previous sections have been utilized to form such equivalence factors which are presented in Figures 5-5 and 5-6. The factors presented here consist of the maximum straight flow rate divided by the corresponding maximum right-turn flow rate. Therefore, the factors should be multiplied by right-turn flow rates to yield equivalent straight flows. As evidenced by the figures, the magnitudes of the factors is significantly different from 1.0 for 2-way stop control but approximately 1.0 for the all-way stop control cases.

Summary

Equivalence factors graphically described in Figures 5-5 and 5-6 provide the capability of equating right-turn and straight movements in terms of their relative maximum flow rates. As presented in these figures, the factors should be multiplied by measured right-turn flow rates in order to convert them to equivalent straight flows. Use of these factors, when evaluating an intersection for possible signal installation, can, in certain volume and control conditions, have a rather significant impact.

RIGHT-TURN BAYS VERSUS RIGHT-TURN LANES

The second of the two questions posed in the introduction to this chapter regarded the use of right-turn bays versus turn lanes. If a turn lane is considered to be a turn bay with a speed change lane, the basic issue becomes, the distance from the intersection to the entrance of the turn bay or lane. This question might be resolved by estimating the length of traffic queue that would typically be created on the intersection approach, because this queue would block entry by right-turn traffic to the turn bay. Prior to developing an experimental testing program, the literature was examined to determine what guidelines are currently used.

Background

Cottrell [Ref 7] extensively examined criteria for the use of right-turn treatments on rural roads in Virginia. His study was largely limited to unsignalized intersections although he conducted a survey of state DOT's in which some information about signalized intersections was also collected. The survey indicated that 16 states used (as of 1980) some type of criteria for choosing a treatment. Parameters most commonly used in the

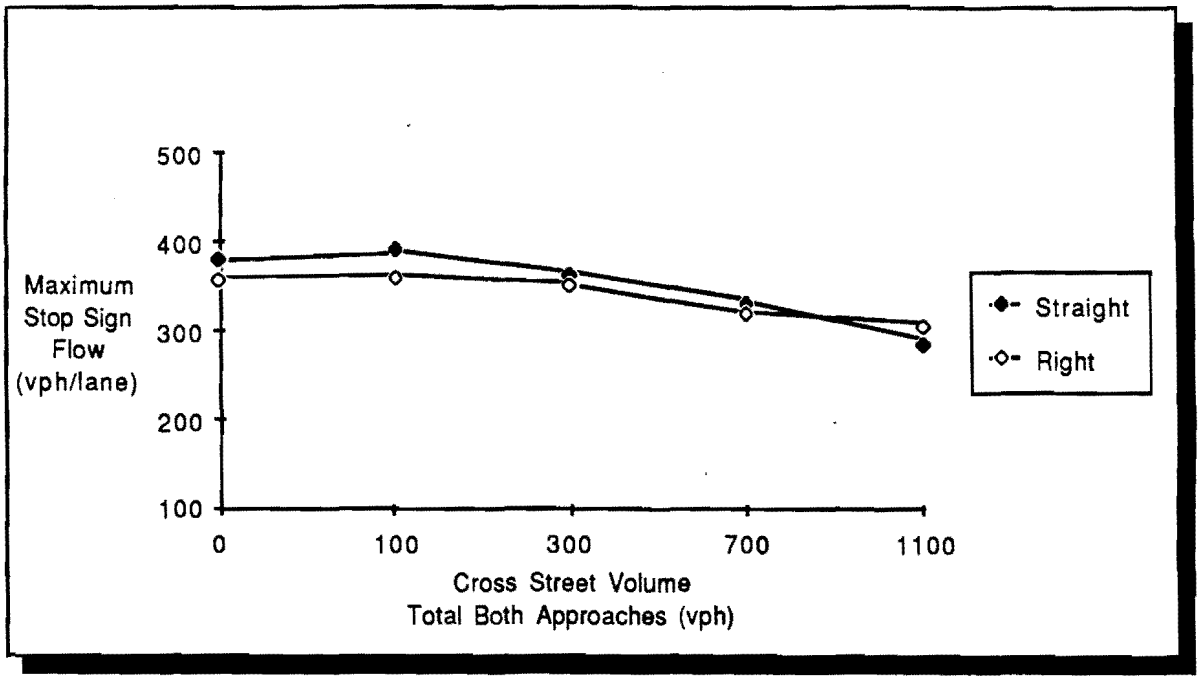


Figure 5-3. Maximum flow rate per lane, 4 X 4 geometry, all-way stop control.

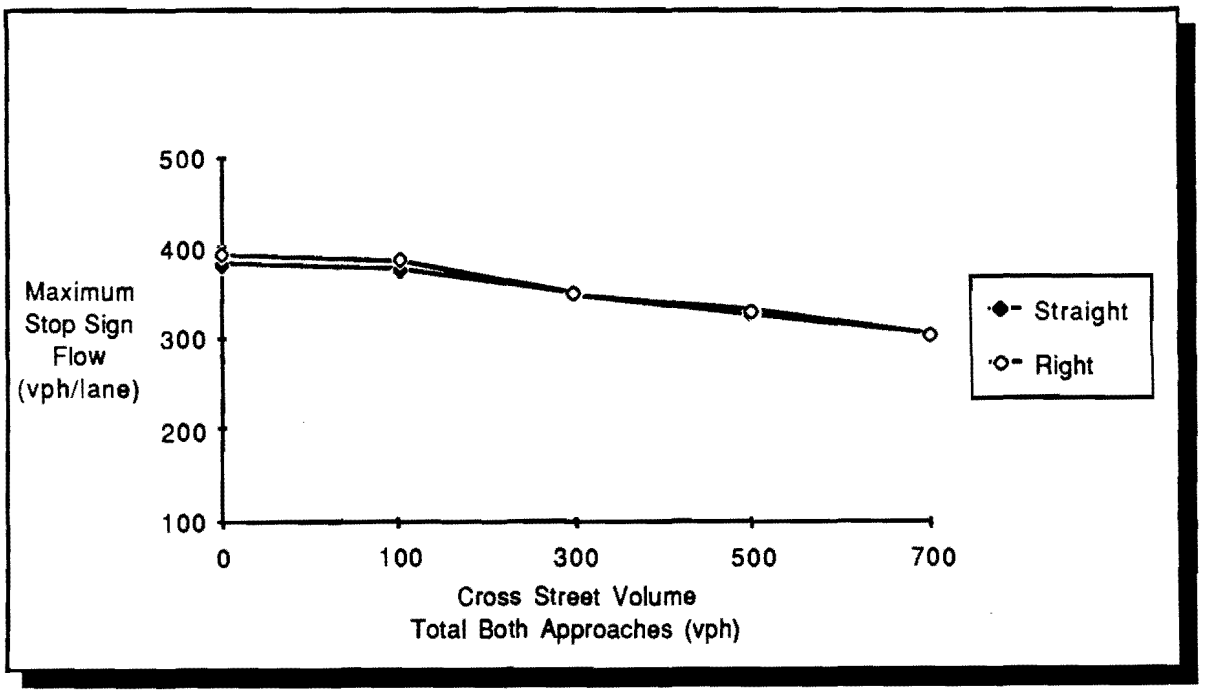


Figure 5-4. Maximum flow rate per lane, 4 X 2 geometry, all-way stop control.

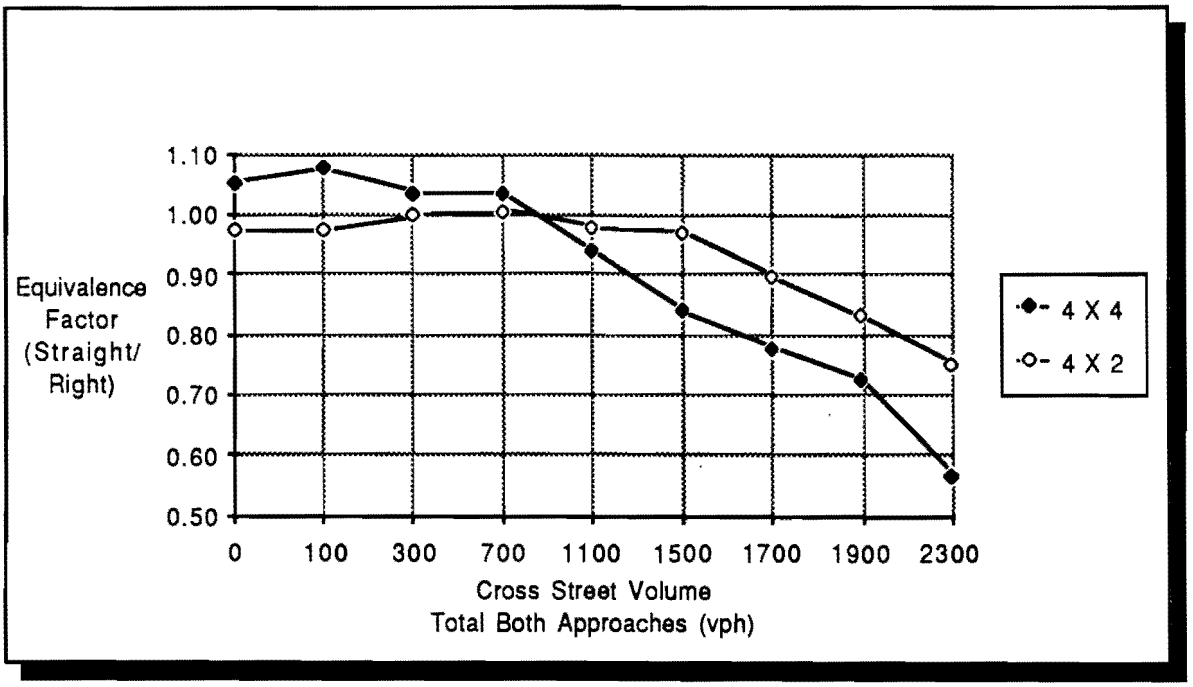


Figure 5-5. Equivalence factors, 2-way stop control, 4 X 4 and 4 X 2 intersection geometry.

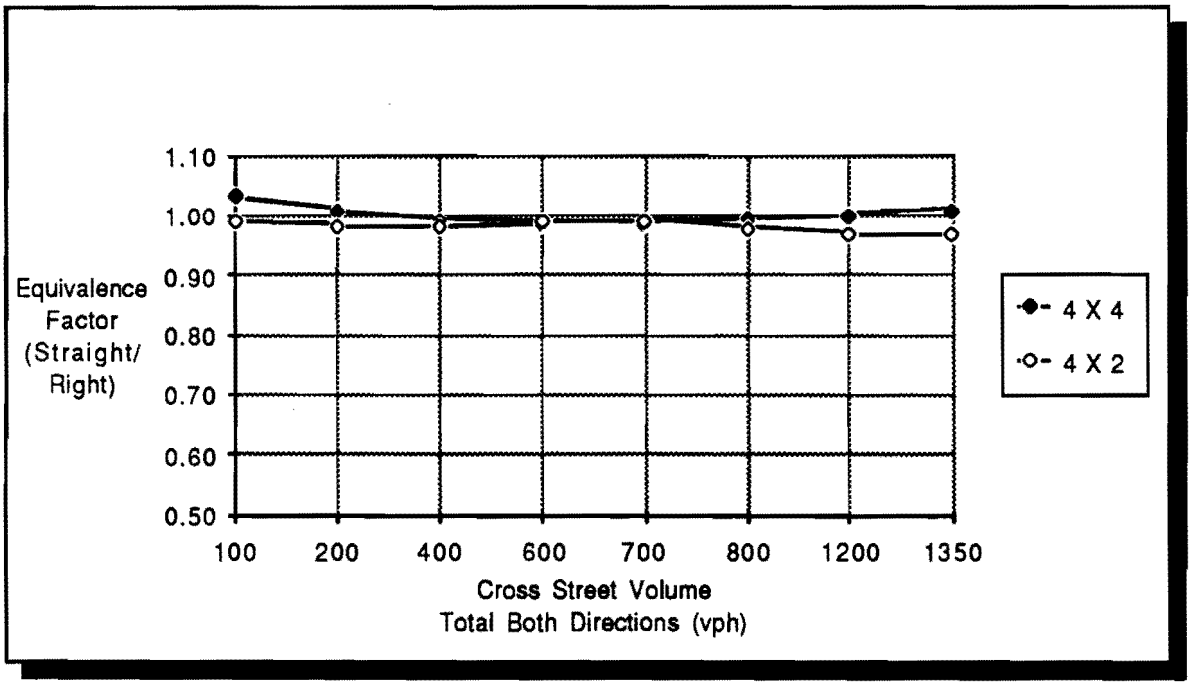


Figure 5-6. Equivalence factor, all-way stop control, 4 X 4 and 4 X 2 intersection geometry.

criteria included right-turn, and straight traffic volumes along with free flow speeds. Safety was a general concern, but no numerical guidelines based on accidents or conflicts were reported. Cottrell also collected field data at 21 rural Virginia intersections and conducted an analysis which led to the development of separate guidelines for two and four lane roadways. The guidelines were based upon total and right-turn peak hour volumes.

Alexander [Ref 6] adopted an economic approach to the development of similar guidelines. Using field data collected at stop-sign controlled intersections, he developed models for predicting delay attributable to right-turn operations. A right-turn lane or bay was considered warranted if the benefits due to reduced delay were greater than the cost of building and maintaining the right-turn treatment. Although accident data was sparse within the sample of 10 field test sites, the three which had right-turn treatments, experienced no relevant accidents while all those without treatments had some right-turn accident experience.

In summary, information located in relevant literature provided a sampling of guidelines for treating right-turn movements. Two significant shortcomings were, however, noted. None of the guidelines provided a general technique for determining an appropriate distance from the intersection stopline back to the entrance of a right-turn bay or lane. Additionally, most of the methodologies were based upon field data collected at stop-sign controlled intersections. Thus, the research effort to be undertaken should deal primarily with the question of how far from the intersection the right-turn lane should begin and should certainly deal with signal controlled intersections.

Experimental Program

Given that a right turn bay or lane may be beneficial, it must be long enough to allow right-turning vehicles to use it without being blocked by through vehicles queued during the red phase. The average queue lengths depend primarily on the through traffic volume and the length of red signal phase.

An experimental testing program was designed using the NETSIM [Ref 8] model to estimate average queue lengths. Traffic volumes of 200 to 800 vehicles per hour per lane were selected as encompassing the usual range of demands. Red signal phases ranging from 10 to 80 seconds were utilized along with a pre-timed signal control scheme.

Optimal cycle lengths were not used for the various demand levels in order to get average queue lengths for long red phases, and to allow queues to completely clear before the beginning of each red phase. Twenty samples were taken at each volume level over the length of the red phase and then averaged to produce a relatively stable statistic. Each simulation run was allowed at least ten minutes of simulation time before collecting data so that system start-up would not adversely affect results.

Average maximum queue lengths as measured from the modeling are presented in Figures 5-7 and 5-8. Graphical results have been presented in two charts in order to enhance the resolution for the lower volume conditions. In all cases the queue lengths tend to increase almost linearly with red time

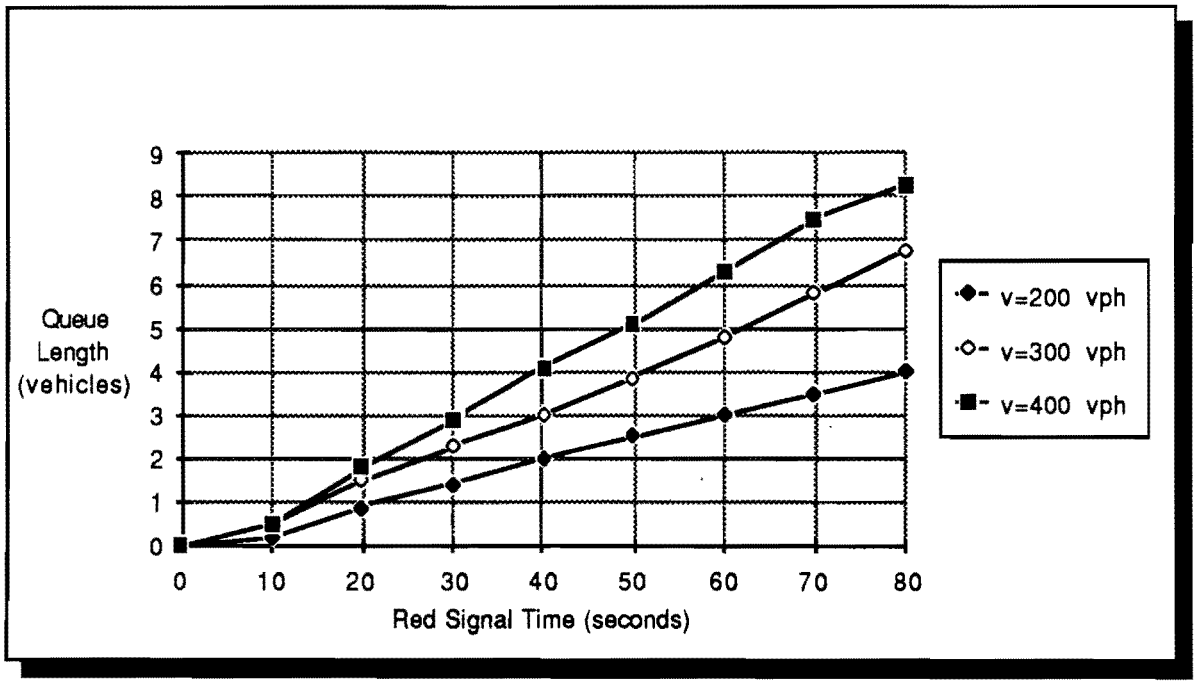


Figure 5-7. Queue lengths versus red signal time (low demand).

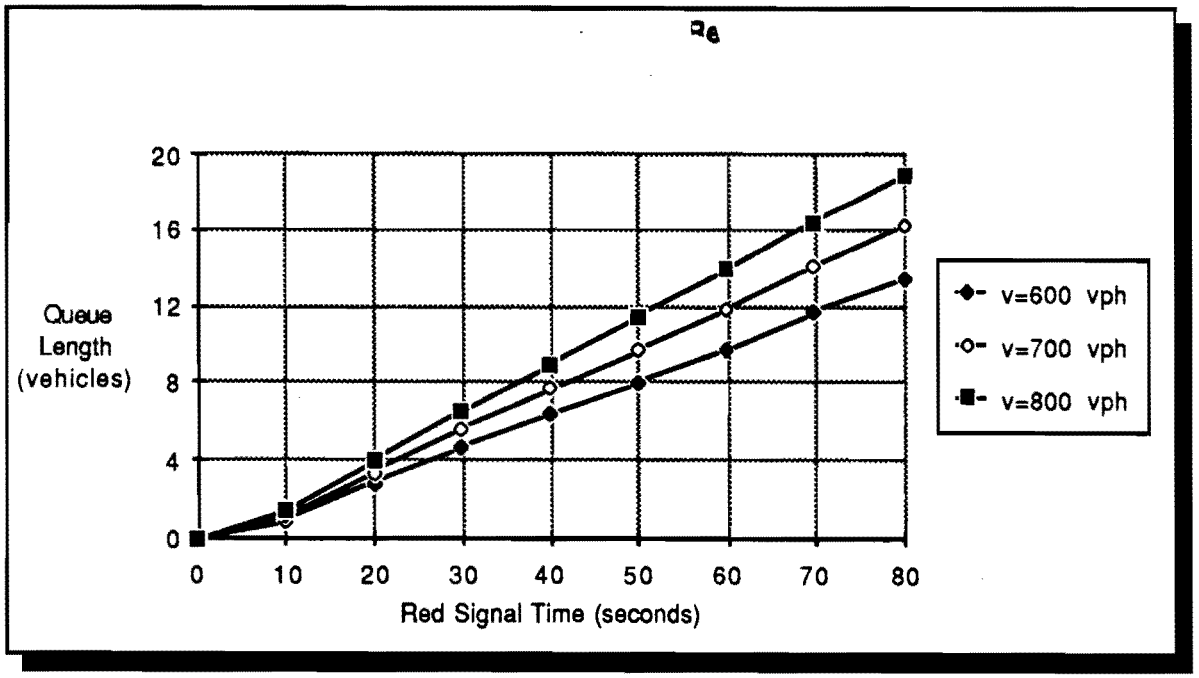


Figure 5-8. Queue lengths versus red signal time (high demand).

Another view of the simulation results is presented in Figures 5-9 and 5-10 in which the average maximum queue lengths of Figures 5-7 and 5-8 are presented in units of feet instead of vehicles. This conversion utilized an assumed vehicle length plus clear space of 19 feet. The ordinates of Figures 5-9 and 5-10 might be interpreted as the minimum distances from the intersection approach stopline at which the opening to a right turn bay or lane must occur if the opening is to not be blocked by through traffic queues.

Figures 5-9 and 5-10 have been termed decision charts since they can assist in making two decisions. First, as noted in the previous paragraph, they can be used to estimate the required distance from the intersection stopline to the entrance to a right-turn bay or lane. Second, they may be used in deciding whether the treatment should be a turn bay or lane. The answer to the bay versus lane question is really dependent upon how one defines each of these treatments. For purposes of the presentation in these figures, a bay has been considered to be a geometric treatment having an entrance less than 100 feet from the intersection stopline. This definition means that a treatment consisting of a short (less than 100 feet) deceleration lane in the vicinity of the intersection would be termed a bay. Although the bay versus lane terminology may not be significant, as a practical matter, if the nose of the right-turn feature must be more than approximately 100 feet from the intersection stopline, it must include a significant length of parallel lane. The 100 feet distance has been used in Figures 5-9 and 5-10 as a, more or less, arbitrary definition of the difference between a right-turn bay and a lane.

Summary

Analyses presented in this chapter have approached two rather different problems which are both related to right-turn operations through at-grade intersections.

1. Equivalence relationships have been developed for converting magnitudes of measured right turn flows into equivalent straight traffic flows. The relationships are based on the relative maximum flow rates of straight and right-turn movements through stop controlled approaches under 2-way and all-way traffic control and 4 X 2 and 4 X 4 intersection lane configurations. The relationships are intended for use in pre-signalization warrant analyses, although they may be used in a variety of other ways.

2. Design guidelines have been provided for determining the required lengths of right-turn bays or lanes. These guidelines have been developed for use at signalized intersections where per lane through traffic volumes range from 200 to 800 vehicles per hour.

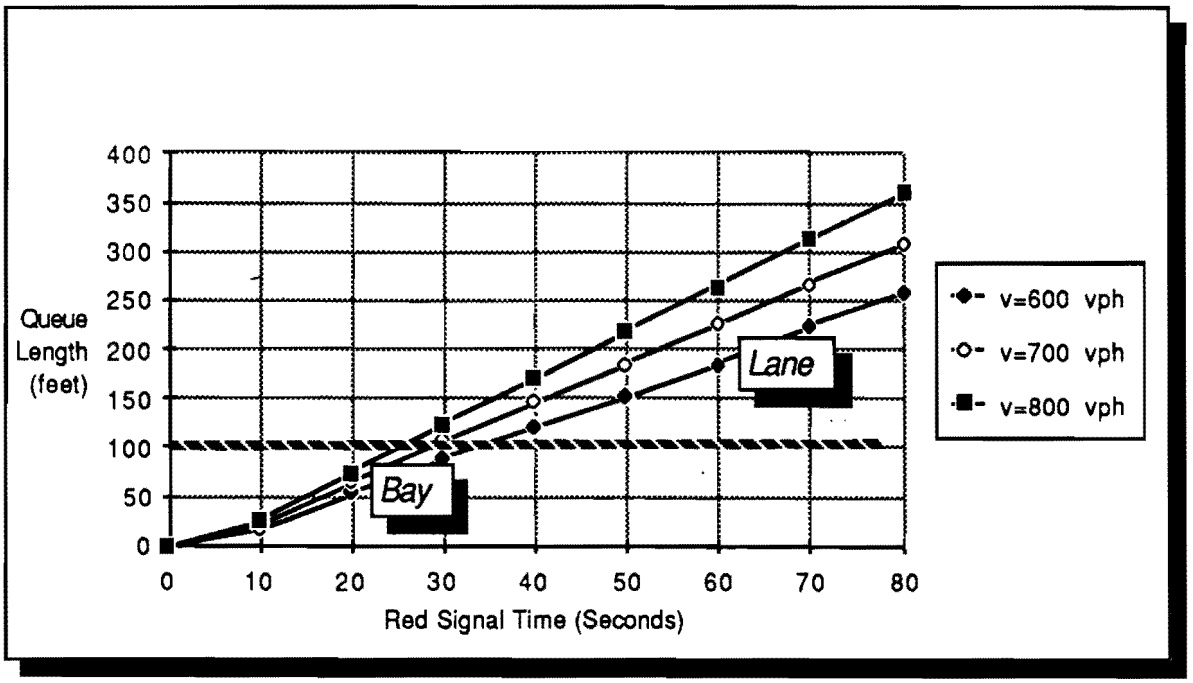


Figure 5-9. Decision chart for turn bays versus lanes (high volumes, 600-800 vph/lane).

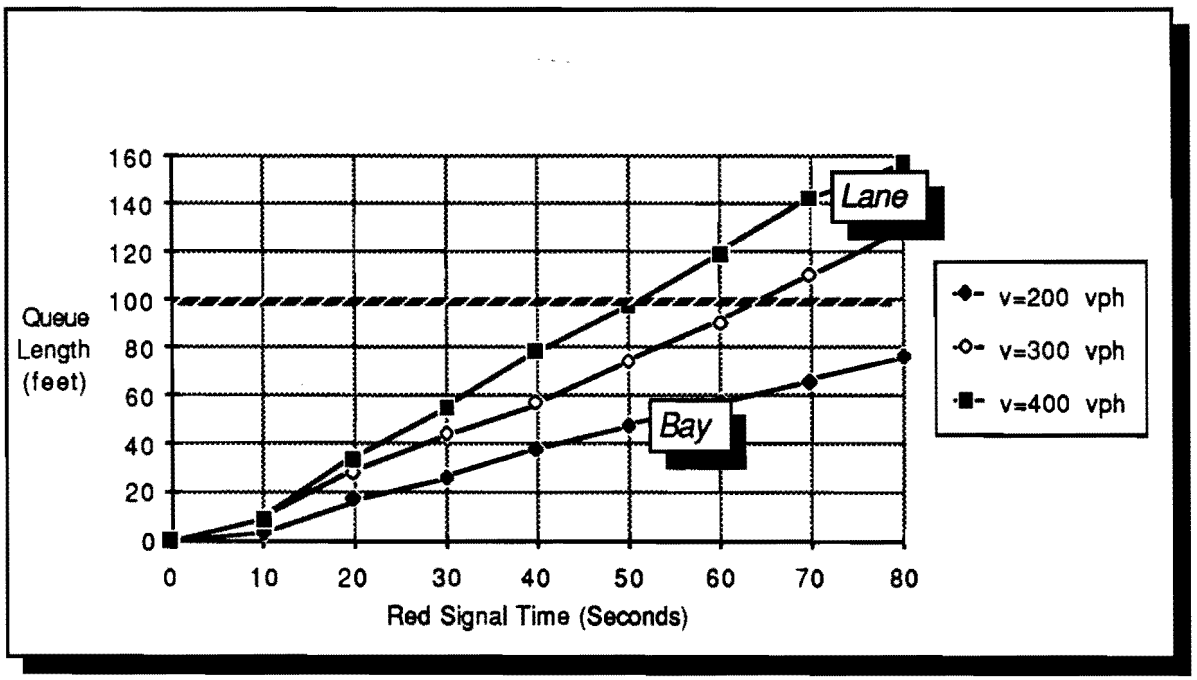


Figure 5-10. Decision chart for bays versus lanes (low volumes, 200-400 vph/lane).

CHAPTER 6. CONCLUSIONS AND RECOMMENDATIONS

Chapters 1 through 4 of this report present documentation of a feasibility study of automated counting of vehicular traffic movements at signalized at-grade intersections. An algorithm was developed for automated counting of traffic movements for the "worst case" of a four-leg, completely unchannelized intersection with multi-lane approaches. The primary technical contribution of this algorithm is that it permits real-time calculation and/or estimation of 12 traffic-movement totals through direct measurement of only eight traffic flows. This is accomplished through the separate estimation of right-turn movements using a process of forecasting the travel time for right-turning vehicles between detectors on inbound and outbound lanes on adjacent intersection legs. The travel time for right-turning vehicles is subject to significant variability. Thus, if the state of the signal controller is monitored, the forecasting procedure is needed only for the right-turns that are made on green-signal indications. The overall accuracy of the counting process is, therefore, as good as any other machine traffic count, with the exception of the right-turns that are executed during green-signal indications.

The algorithm has been implemented using totally off-the-shelf hardware. Requirements include a microcomputer and an interface device that accommodates at least nine TTL-compatible digital input channels. The microprocessor must have sufficient speed to sample the nine, or more, input channels approximately 100 times per second. All instruction code has been written in ANSI Standard 3.2 FORTRAN, with the exception of calls to the interface device which are hardware specific.

Experience gained through the development and testing program have led to the following conclusions and recommendations:

1. Automated counting of vehicular traffic at signalized intersections is feasible using currently-available, off-the-shelf hardware.
2. Vehicle counts are the desired end product of most intersection traffic counting efforts. The accuracy of automated counting can be significantly enhanced through direct sensing of vehicle passage; therefore, detectors to be utilized for automated counting should sense the passage of vehicles rather than axles.
3. Considering cost, ease of installation, reliability, and accuracy the most desirable detection system investigated thus far for use in automated turning-movement counting is an infra-red light beam system operating in the reflex mode.
4. Effort required for installation of eight detection devices near an intersection is the greatest impediment to routine use of the automated turning-movement counting technique described herein. This effort could be significantly reduced if low-cost, wireless transmission of detector actuations could be utilized. Development of hardware for this application is recommended.

REFERENCES

1. Jeffreys, M., and M. Norman, "On Finding Realistic Turning flows at Road Junctions," Traffic Engineering and Control, Vol. 18, No. 1, January 1977, pp. 19-21, 25; Addendum, April 1977, p. 207.
2. Mekky, A., "On Estimating Turning Flows at a Road Junction," Traffic Engineering and Control, Vol. 18, No. 1, January 1977, pp. 19-21, 25; Addendum, April 1977, p. 207.
3. Van Zuylen, H.J., "The Estimation of Turning Flows on a Junction," Traffic Engineering and Control, Vol. 20, No. 11, November 1979, pp. 539-541.
4. Norman, M., and N. Hoffman, "Non-Iterative Methods for Generating a Realistic Turning Flow Matrix for a Junction," Traffic Engineering and Control, Vol. 20, No. 12, December 1979, pp. 587-589.
5. Hauer, E., E. Pagitsas, and B.T. Shin, "Estimation of Turning Flows from Automatic Counts," Transportation Research Record 795, 1981, pp. 1-7.
6. Alexander, Michael H., "Development of An Economic Warrant for the Construction of Right-Turn Deceleration Lanes," Purdue University, Joint Research Project, Report No. 12, May 1970.
7. Cottrell, B.H., Jr., "The Development of Criteria for the Treatment of Right-Turn Movements on Rural Roads," Virginia Highway and Transportation Research Council, Virginia Department of Highways and Transportation, the University of Virginia, 1981.
8. "Traffic Network Analysis with NET S IM - A User Guide," U.S. Department of Transportation, Federal Highway Administration, Washington, D.C., January 1980.
9. Kraft, Merwyn A. and C. Thomas Van Vechten, "An Investigation of Traffic Behavior at Signals with Red, Yellow, and Green Arrows," Traffic Research Section, D.C. Department of Highways and Traffic, 1972.
10. Lee, Clyde E., Glenn E. Grayson, Charlie R. Copeland, Jeff W. Miller, Thomas W. Rioux, and Vivek S. Savur, "The TEXAS Model for Intersection Traffic - User's Guide," Center for Transportation Research Report No. 184-3, State Department of Highways and Public Transportation, Federal Highway Administration, The University of Texas at Austin, July 1977.
11. Mamlouk, Michael S., "Right Turn on Red: Utilization and Impact," Purdue University, Report No. JHRP-76-17, 1976.
12. May, Ronald L., "RTOR: Warrants and Benefits," Purdue University, Report No. JHRP-74-14, 1974.

APPENDIX A. LISTING OF FORTRAN PROGRAM FOR AUTOMATED TURNING MOVEMENT COUNTING (2-PHASE SIGNAL CONTROL)

```
      PROGRAM CAL
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      CHARACTER*79 TMDAT
      COMMON / SKEC / TMDAT(15)
      OPEN(NIO,FILE='CON')
C
C ----- DISPLAY SKETCH OF DETECTOR LOCATIONS & CONFIRM CONNECTION TO DAC
C
      CALL CDC
C
C ----- RUN CALIBRATION ROUTINE
C
      CALL CALSUB
      END
      BLOCK DATA
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      COMMON / DAC / ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      INTEGER*2 ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      DATA NIO,NTMRC/10,-1/
      DATA ((IDET(I,J),J=1,2),I=1,8)/1,5,0,0,3,8,0,0,0,0,6,4,7,2,0,0/,
     1     NUM/8*0/,SUM/8*0./,SUMSQ/8*0./,W1/8*1./,W2/8*19./
      DATA ADAPT/0/,DEVICE/8/,STAT/0/,HNDSHK/0/
      END
```

```
      SUBROUTINE TI
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       RAWVAL,ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,NTMRC,ITMROW(20),ITMCOL(20),INTERV,INTTOT
      INTEGER ST RT TM
      CHARACTER*79 TMDAT(15),ERRMSG
      CHARACTER FMT1*6,AFMT*5
      PARAMETER(N1=1,FMT1='(20I4)',AFMT='(A79)')
C
C ----- FILE WITH SKETCH DATA
C
      OPEN(4,FILE='TM.DAT')
      REWIND 4
C
C ----- READ NUMBER OF DETECTORS IN SKETCH
C
      READ(4,FMT1)NTMRC
C
C ----- READ ROW NUMBER FOR EACH DETECTOR ID IN THE SKETCH
C
      READ(4,FMT1)(ITMROW(I),I=N1,NTMRC)
C
C ----- READ COLUMN NUMBER FOR EACH DETECTOR ID IN THE SKETCH
      READ(4,FMT1)(ITMCOL(I),I=N1,NTMRC)
      NTMDAT=N1
      DO 50 I=N1,NTMRC
      ITMROW(I)=ITMROW(I)+N1
   50 CONTINUE
  100 CONTINUE
C
C ----- READ DATA FOR SKETCH
C
      READ(4,AFMT,END=200)TMDAT(NTMDAT)
      NTMDAT=NTMDAT+N1
      GO TO 100
  200 CONTINUE
      NTMDAT=NTMDAT-N1
      OPEN(NIO,FILE='CON')
C
C ----- CLEAR SCREEN
C
      WRITE(NIO,'(2A)')CHAR(27),'[2J'
C
C ----- WRITE SKETCH TO TERMINAL
C
      DO 250 I=N1,NTMDAT
      NC=ILNB(TMDAT(I))
      WRITE(NIO,AFMT)TMDAT(I)
  250 CONTINUE
      DO 300 I=N1,NTMRC
```

```
C
C ----- PUT DETECTOR ID NUMBERS ON SKETCH (HIGHLITED)
C
      CALL TMNUMIO(ITMROW(I),ITMCOL(I),I,2,NIO)
  300 CONTINUE
C
C ----- DEFINE PROMPT WINDOW TO LEAVE 1 BLANK LINE BELOW SKETCH
C
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+2,'H'
      WRITE(NIO,'(A)')'Press <Enter> to confirm detector connections.'
      READ(NIO,'(A)')
      DO 400 I=N1,NTMRC
C
C ----- CHANGE DETECTOR ID NUMBERS ON SKETCH TO NORMAL BRIGHTNESS
C
      CALL TMNUMIO(ITMROW(I),ITMCOL(I),I,1,NIO)
  400 CONTINUE
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+2,'H'
      WRITE(NIO,'(2A)')CHAR(27),'[K'
C     WRITE(NIO,'(2A)')CHAR(27),'[B'
      WRITE(NIO,'(2A)')CHAR(27),'[K'
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+2,'H'
      WRITE(NIO,'(A)')'Please specify the interval between '//
     1     'successive count totals (in minutes).'
      IFROM=1
  450 CONTINUE
      READ(NIO,*,ERR=1010,IOSTAT=IERR)INTERV
      IF(IMIN1.LE.0)THEN
        ERRMSG='TIME INTERVAL MUST BE POSITIVE.  PLEASE RE-ENTER DATA.'
        GO TO 1020
      END IF
      WRITE(NIO,'(2A,I2.2,3A)')CHAR(27),'[',NTMDAT+2,'H'
      WRITE(NIO,'(2A)')CHAR(27),'[K'
      WRITE(NIO,'(2A)')CHAR(27),'[K'
      WRITE(NIO,'(2A)')CHAR(27),'[K'
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+2,'H'
      WRITE(NIO,'(A)')'Please specify the total time for which '//
     1     'counts are to be made.'
      IFROM=2
  490 CONTINUE
      READ(NIO,*,ERR=1010,IOSTAT=IERR)INTTOT
      IF(IMIN2.LT.IMIN1)THEN
        ERRMSG='TOTAL TIME MUST NOT BE LESS THAN TIME INTERVAL.'//
     1         ' PLEASE RE-ENTER DATA.'
        GO TO 1020
      END IF
      GO TO 4000
 1010 CONTINUE
      ERRMSG='ERROR IN KEYIN.  PLEASE RE-ENTER DATA.'
 1020 CONTINUE
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+3,'H'
      WRITE(NIO,'(3A)')CHAR(27),'[K'
```

```
      WRITE(NIO,'(3A)')CHAR(27),'[K'
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+4,'H'
      WRITE(NIO,'(A)')ERRMSG
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+3,'H'
      GO TO (450,490) IFROM
 4000 CONTINUE
      RETURN
      END
      FUNCTION ILNB(LINE)
C
C   THIS FUNCTION SETS <ILNB> TO THE POSITION NUMBER OF THE
C   LAST (RIGHTMOST) NON-BLANK CHARACTER IN THE CHARACTER
C   VARIABLE <LINE>.
C
      CHARACTER*(*) LINE
      PARAMETER(N1=1)
      DO 100 I=LEN(LINE),N1,-N1
      ILNB=I
      IF(LINE(I:I).NE.' ')RETURN
  100 CONTINUE
      ILNB=0
      RETURN
      END
      SUBROUTINE TMNUMIO(IROW,ICOL,INUM,I,NIO)
C
C ----- THIS SUBROUTINE WRITES A SINGLE NUMBER TO A SPECIFIC ROW AND COLUMN
C ----- THE NUMBER CAN BE NORMAL INTENSITY, BRIGHT OR BLINKING
C
C   IROW - INTEGER ROW
C   ICOL - INTEGER COLUMN
C   INUM - NUMBER TO BE WRITTEN
C   I - METHOD OF DISPLAY:
C       1 = NORMAL INTENSITY
C       2 = BRIGHT
C       3 = BLINKING
C   NIO - UNIT NUMBER THAT IS OPENED FOR I/O TO THE TERMINAL
C
C     CHARACTER*1 GR(3)
C     DATA (GR(I),I=1,3) /'0','1','5'/
      INTEGER GR(3)
      DATA (GR(I),I=1,3)/7,15,135/
      CALL SCP(NIO)
C     WRITE(NIO,'(2A,I2.2,A,I2.2,A,4A,I1,4A)')
C    1       CHAR(27),'[',IROW,';',ICOL,'H',
C    2       CHAR(27),'[',GR(I),'m',INUM,CHAR(27),'[',GR(1),'m'
       CALL DINCH(CHAR(INUM+ICHAR('0')),GR(I),IROW,ICOL,0)
      CALL RCP(NIO)
      RETURN
      END
      SUBROUTINE SCP(NIO)
C
C ----- THIS SUBROUTINE SAVES THE CURRENT CURSOR POSITION (ROW AND COLUMN).
```

```
C ----- THE CURSOR CAN BE MOVED TO THE MOST CURRENT SAVED POSITION BY
C ----- CALLING SUBROUTINE RCP.
C   NIO - UNIT NUMBER THAT IS OPEN FOR I/O TO CONSOLE
C
      WRITE(NIO,'(2A)')CHAR(27),'[s'
      RETURN
      END
      SUBROUTINE RCP(NIO)
C
C ----- THIS SUBROUTINE RESTORES THE CURSOR TO THE MOSE CURRENT SAVED POWITION.
C ----- CURSOR POSITION CAN BE SAVED BY CALLING SUBROUTINE SCP.
C   NIO - UNIT NUMBER THAT IS OPEN FOR I/O TO CONSOLE.
C
      WRITE(NIO,'(4A)')CHAR(27),'[u',CHAR(27),'[A'
      RETURN
      END
```

```
FUNCTION ILNB(LINE)
C
C   THIS FUNCTION SETS <ILNB> TO THE POSITION NUMBER OF THE
C   LAST (RIGHTMOST) NON-BLANK CHARACTER IN THE CHARACTER
C   VARIABLE <LINE>.
C
      CHARACTER*(*) LINE
      PARAMETER(N1=1)
      DO 100 I=LEN(LINE),N1,-N1
      ILNB=I
      IF(LINE(I:I).NE.' ')RETURN
  100 CONTINUE
      ILNB=0
      RETURN
      END
```

```
      SUBROUTINE RNG(IXL,IXU,IYL,IYU,IXLI,IXUI,IXLF,IXUF,FX,DX)
      CHARACTER*5 TEXT
      CALL VL(IXLI,IYL,IYU,129,0)
      CALL VL(IXUI,IYL,IYU,129,0)
      IXLF=IXLI
      IXUF=IXUI
      CALL DINCH('Up, dn., left & rt. arrows make coarse',7,22,0,0)
      CALL DINCH('window changes. Shift-arrows make fine',7,23,0,0)
      CALL DINCH('changes. Press <Enter> when done.',7,24,0,0)
      CALL DINCH('WINDOW START:',7,19,21,0)
      CALL DINCH('WINDOW END:',7,20,23,0)
      GO TO 2000
 1000 CONTINUE
      CALL CHKBD(IKBD)
      IF(IKBD.EQ.0)GO TO 1000
      IF(IKBD.GT.56)GO TO 1500
      IF(IKBD.EQ.54)THEN
C
C ----- SHIFT-<RIGHT ARROW>
C
        CALL VL(IXUF,IYL,IYU,129,0)
        IXUF=IXUF+1
        CALL VL(IXUF,IYL,IYU,129,0)
        CALL VL(IXLF,IYL,IYU,129,0)
        IXLF=IXLF+1
        CALL VL(IXLF,IYL,IYU,129,0)
        GO TO 2000
      END IF
      IF(IKBD.EQ.52)THEN
C
C ----- SHIFT-<LEFT ARROW>
C
        CALL VL(IXUF,IYL,IYU,129,0)
        IXUF=IXUF-1
        CALL VL(IXUF,IYL,IYU,129,0)
        CALL VL(IXLF,IYL,IYU,129,0)
        IXLF=IXLF-1
        CALL VL(IXLF,IYL,IYU,129,0)
        GO TO 2000
      END IF
      IF(IKBD.EQ.56)THEN
C
C ----- SHIFT-<UP ARROW>
C
        CALL VL(IXUF,IYL,IYU,129,0)
        IXUF=IXUF+1
        CALL VL(IXUF,IYL,IYU,129,0)
        CALL VL(IXLF,IYL,IYU,129,0)
        IXLF=IXLF-1
        CALL VL(IXLF,IYL,IYU,129,0)
        GO TO 2000
      END IF
```

```
      IF(IKBD.EQ.50)THEN
C
C ----- SHIFT-<DOWN ARROW>
C
        CALL VL(IXUF,IYL,IYU,129,0)
        IXUF=IXUF-1
        CALL VL(IXUF,IYL,IYU,129,0)
        CALL VL(IXLF,IYL,IYU,129,0)
        IXLF=IXLF+1
        CALL VL(IXLF,IYL,IYU,129,0)
        GO TO 2000
      END IF
C
C ----- CHECK FOR <ENTER>
C
      IF(IKBD.EQ.13)GO TO 9990
      GO TO 1000
 1500 CONTINUE
      IF(IKBD.EQ.77)THEN
C
C ----- <RIGHT ARROW>
C
        CALL VL(IXUF,IYL,IYU,129,0)
        IXUF=IXUF+5
        CALL VL(IXUF,IYL,IYU,129,0)
        CALL VL(IXLF,IYL,IYU,129,0)
        IXLF=IXLF+5
        CALL VL(IXLF,IYL,IYU,129,0)
        GO TO 2000
      END IF
      IF(IKBD.EQ.75)THEN
C
C ----- <RIGHT ARROW>
C
        CALL VL(IXUF,IYL,IYU,129,0)
        IXUF=IXUF-5
        CALL VL(IXUF,IYL,IYU,129,0)
        CALL VL(IXLF,IYL,IYU,129,0)
        IXLF=IXLF-5
        CALL VL(IXLF,IYL,IYU,129,0)
        GO TO 2000
      END IF
      IF(IKBD.EQ.72)THEN
C
C ----- <UP ARROW>
C
        CALL VL(IXUF,IYL,IYU,129,0)
        IXUF=IXUF+5
        CALL VL(IXUF,IYL,IYU,129,0)
        CALL VL(IXLF,IYL,IYU,129,0)
        IXLF=IXLF-5
        CALL VL(IXLF,IYL,IYU,129,0)
```

```
        GO TO 2000
      END IF
      IF(IKBD.EQ.80)THEN
C
C ----- <DOWN ARROW>
C
        CALL VL(IXUF,IYL,IYU,129,0)
        IXUF=IXUF-5
        CALL VL(IXUF,IYL,IYU,129,0)
        CALL VL(IXLF,IYL,IYU,129,0)
        IXLF=IXLF+5
        CALL VL(IXLF,IYL,IYU,129,0)
        GO TO 2000
      END IF
      GO TO 1000
 2000 CONTINUE
      X=FX+(IXLF-IXL)*DX
      WRITE(TEXT,'(F5.2)')X
      CALL DINCH(TEXT,3,19,35,0)
      X=FX+(IXUF-IXL)*DX
      WRITE(TEXT,'(F5.2)')X
      CALL DINCH(TEXT,3,20,35,0)
      GO TO 1000
 9990 CONTINUE
      RETURN
      END
c     SUBROUTINE BAR(IX1,IY1,IX2,IY2,IA,IP)
c
C ----- THIS SUBROUTINE DRAWS INDIVIDUAL VERTICAL BARS FOR A BAR CHART
C
C   IX1,IY1 - PIXEL NUMBER OF LOWER LEFT CORNER OF BAR
C   IX2,IY2 -                 UPPER RIGHT
C              0 THRU 319 FOR X       0 THRU 199 FOR Y
C              (0,0) IS UPPER LEFT OF SCREEN
C              (319,199) IS LOWER RIGHT OF SCREEN
C   IA - COLOR/ATTRIBUTE
C   IP DISPLAY PAGE
C
c     DO 1000 J=IX1,IX2
c     CALL VL(J,IY1,IY2,IA,NP)
c1000 CONTINUE
c     RETURN
c     END
      SUBROUTINE SC(NREV)
C
C ----- THIS SUBROUTINE DRAWS A CHART ON THE SCREEN TO HELP REVISE
C ----- A WINDOW
C   NREV - INDEX OF THE WINDOW TO BE REVISED
C
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
```

```
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      CHARACTER TEXT*10
      DX=5./56.
      CALL STMODE(4)
      IT2=20
      DO 100 I=1,15,2
      WRITE(TEXT,'(I2)')I
      IT1=I-1
      CALL HL(IT2+1,IT2+2,(IT1*8)+3,3,0)
      IF(I.NE.15)CALL HL(IT2+1,IT2+2,(I*8)+3,3,0)
      CALL DINCH(TEXT(:2),3,IT1,0,0)
  100 CONTINUE
      IT1=IT1+1
      CALL VL(IT2,0,(IT1*8)+3,3,0)
      IT1=IT1+1
      IT3=121
      IT4=IT3+1
      IT5=IT2+1
      J=1
      DO 200 I=0,20,5
      X=I
      WRITE(TEXT,'(F4.1)')X
      IF(X.LT.10)THEN
        IO=1
      ELSE
        IO=0
      END IF
      CALL DINCH(TEXT(:4),3,IT1,J-IO,0)
      J=J+7
      IF(I.NE.0)CALL VL(IT2,IT3,IT4,3,0)
      IT2=IT2+28
      IF(I.LT.20)CALL VL(IT2,IT3,IT4,3,0)
      IT2=IT2+28
  200 CONTINUE
      IT1=IT1-1
      CALL HL(IT5,IT2-14,(IT1*8)+3,3,0)
      CALL DINCH('TIME (SEC.)',3,IT1+2,8,0)
      IXLI=NINT(W1(NREV)/DX)+20
      IXUI=NINT(W2(NREV)/DX)+20
      DO 300 I=1,NUM(NREV)
      ICX=NINT(TIMEE(I,NREV)/DX)+20
      ICY=(I-1)*8+3
      CALL CROSS(ICX,ICY,2,0)
  300 CONTINUE
      CALL RNG(20,200,1,119,IXLI,IXUI,IXLF,IXUF,0.0,DX)
      W1(NREV)=(IXLF-20)*DX
      W2(NREV)=(IXUF-20)*DX
      RETURN
      END
      SUBROUTINE CROSS(IX,IY,IA,IP)
C
```

```
C ----- THIS SUBROUTINE DRAWS A CROSS ON A DISPLAY PAGE
C
C   IX,IY - PIXEL NUMBER OF THE CENTER OF THE CROSS
C           (0,0) IS UPPER LEFT OF SCREEN
C           (319,199) IS LOWER RIGHT OF SCREEN
C   IA - COLOR/ATTRIBUTE
C   IP - DISPLAY PAGE
C
      INTEGER*2 IPXY(18)
      DATA IXL,IYL/-1,-1/
      IF(IX.EQ.IXL)GO TO 100
      IXL=IX
      IPXY(1)=IX-2
      IPXY(3)=IX-1
      IPXY(5)=IX
      IPXY(7)=IX+1
      IPXY(9)=IX+2
      IPXY(11)=IX
      IPXY(13)=IX
      IPXY(15)=IX
      IPXY(17)=IX
  100 CONTINUE
      IF(IY.EQ.IYL)GO TO 200
      IYL=IY
      IPXY(2)=IY
      IPXY(4)=IY
      IPXY(6)=IY
      IPXY(8)=IY
      IPXY(10)=IY
      IPXY(12)=IY+1
      IPXY(14)=IY-1
      IPXY(16)=IY+2
      IPXY(18)=IY-2
  200 CONTINUE
      CALL DINPT(IPXY,IA,9,IP)
      RETURN
      END
      SUBROUTINE VL(IX,IY1,IY2,IA,IP)
C
C ----- THIS SUBROUTINE DRAWS VERTICAL LINES ON A DISPLAY PAGE
C
C   IX - HORIZONTAL PIXEL NUMBER FOR LINE
C   IY1 - TOP PIXEL NUMBER FOR LINE
C   IY2 - BOTTOM
C         (0,0) IS UPPER LEFT OF SCREEN
C         (319,199) IS LOWER RIGHT OF SCREEN
C   IA - COLOR/ATTRIBUTE
C   IP - DISPLAY PAGE
C
      INTEGER*2 IPXY(400)
      DATA NP1,IXL,IY1L,IY2L / 0,-1,2*0 /
      IF((IY1.EQ.IY1L) .AND. (IY2.EQ.IY2L))GO TO 200
```

```
      IT1=IY2-IY1+1
      NP1=IT1
      IT1=IT1+IT1
      IY1L=IY1
      IY2L=IY2
      IT2=IY1
      DO 100 I=2,IT1,2
      IPXY(I)=IT2
      IT2=IT2+1
  100 CONTINUE
  200 CONTINUE
      IF(IX.EQ.IXL)GO TO 600
      IXL=IX
      DO 400 I=1,IT1,2
      IPXY(I)=IX
  400 CONTINUE
  600 CONTINUE
      CALL DINPT(IPXY,IA,NP1,IP)
      RETURN
      END
      SUBROUTINE HL(IX1,IX2,IY,IA,IP)
C
C ----- THIS SUBROUTINE DRAWSHORIZONTAL LINES ON A DISPLAY PAGE
C
C   IX1 - LEFTMOST PIXEL NUMBER FOR LINE
C   IX2 - RIGHTMOST
C   IY - VERTICAL PIXEL NUMBER FOR LINE
C        (0,0) IS UPPER LEFT OF SCREEN
C        (319,199) IS LOWER RIGHT OF SCREEN
C   IA - COLOR/ATTRIBUTE
C   IP - DISPLAY PAGE
C
      INTEGER*2 IPXY(640)
      DATA NP1,IX1L,IX2L,IYL / 0,2*0,-1/
      IF((IX1.EQ.IX1L) .AND. (IX2.EQ.IX2L))GO TO 200
      IT1=IX2-IX1+1
      NP1=IT1
      IT1=IT1+IT1
      IX1L=IX1
      IX2L=IX2
      IT2=IX1
      DO 100 I=1,IT1,2
      IPXY(I)=IT2
      IT2=IT2+1
  100 CONTINUE
  200 CONTINUE
      IF(IY.EQ.IYL)GO TO 600
      IYL=IY
      DO 400 I=2,IT1,2
      IPXY(I)=IY
  400 CONTINUE
  600 CONTINUE
```

```
CALL DINPT(IPXY,IA,NP1,IP)
RETURN
END
```

```
      PROGRAM COUNT
      CHARACTER ERRMSG*79
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      INTEGER*2 IH,IM,IS,IHUN
      COMMON /B/ RES(9),RES1(9)
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      LOGICAL RES,RES1,BTEST
      COMMON / DAC / ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      INTEGER*2 ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      LOGICAL TRTF,NSGRN,NSGRNP
      EQUIVALENCE(RES1(9),TRTF)
      DATA IC / 0 /
      OPEN(NIO,FILE='CON')
C
C ----- DISPLAY DETECTOR SKETCH & CONFIRM CONNECTION TO DAC
C
      CALL CDC
C
C ----- DISPLAY GREEN STATUS AND CONFIRM CONNECTION
C
      CALL CGS
C-----PROMPT FOR THE COUNT INTERVAL, TOTAL COUNT TIME
      CALL TI
      INTERV=INTERV*60
      INTTOT=INTTOT*60
      IT1=NTMDAT+2
C
C ----- TRANSITION TIME
C
      CALL ERNLNS(IT1,6,IT1,NIO)
      WRITE(NIO,'(A)')'Please enter the time for transition '//
     1                'between phases (in seconds).'
 1005 CONTINUE
      READ(NIO,*,ERR=1010,IOSTAT=IERR)TRTIME
      IF(TRTIME.LT.0.0)THEN
        ERRMSG='TRANSITION TIME MUST BE POSITIVE. PLEASE RE-ENTER DATA.'
        GO TO 1020
      END IF
      GO TO 1040
 1010 CONTINUE
      ERRMSG='ERROR IN KEYIN. PLEASE RE-ENTER DATA.'
 1020 CONTINUE
      CALL ERNLNS(NTMDAT+3,2,NTMDAT+4,NIO)
      WRITE(NIO,'(A)')ERRMSG
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+3,'H'
      GO TO 1005
 1040 CONTINUE
```

```
C-----CALL THE SUBROUTINE THAT PROVIDES RIGHT TURN WINDOW DATA
      CALL INSET
      CALL ERNLNS(IT1,7,IT1,NIO)
C
C ----- DISPLAY HEADER FOR COUNTS
C
      ICDR=IT1+2
      CALL DINCH('NORTHBOUND',7,ICDR,6,0)
      CALL DINCH('WESTBOUND',7,ICDR,27,0)
      CALL DINCH('SOUTHBOUND',7,ICDR,46,0)
      CALL DINCH('EASTBOUND',7,ICDR,67,0)
      ICDR=ICDR+1
      CALL DINCH('L     S     R',7,ICDR,5,0)
      CALL DINCH('L     S     R',7,ICDR,25,0)
      CALL DINCH('L     S     R',7,ICDR,45,0)
      CALL DINCH('L     S     R',7,ICDR,65,0)
C
C ----- DISPLAY ZERO COUNTS
C
      ICDR=ICDR+1
      CALL DCCZ(ICDR)
      CALL ERNLNS(IT1,1,IT1,NIO)
      WRITE(NIO,'(A)')'Press any key when you want to start counting.'
 2000 CONTINUE
C
C ----- MONITOR SIGNAL AND DETECTORS
C
      CALL BINS(ADAPT,DEVICE,HNDSHK,RAWVAL, STAT)
      CALL DGS(BTEST(RAWVAL,9))
      CALL DDS(RAWVAL)
      CALL CHKBD(IKBD)
      IF(IKBD.EQ.0)GO TO 2000
      CALL GETTIM(IH,IM,IS,IHUN)
      ITIME=IS+IM*60+IH*3600
      TIME=IHUN
      TIME=TIME*.01
      TIME=TIME+ITIME
      TN=INTERV+TIME
      TNTOT=INTTOT+TIME
      CALL ERNLNS(IT1,1,IT1,NIO)
      IF(IH.GT.9)THEN
        WRITE(NIO,'(A,I2,A,3(I2.2,A))')'Counting is in progress. '//
     1     'Counting started at ',ih,':',im,':',is,'.',ihun,'.'
      ELSE
        WRITE(NIO,'(A,I1,A,3(I2.2,A))')'Counting is in progress. '//
     1     'Counting started at ',ih,':',im,':',is,'.',ihun,'.'
      END IF
      CALL BINS(ADAPT,DEVICE,HNDSHK,RAWVAL, STAT)
      NSGRN=BTEST(RAWVAL,9)
      NSGRNP=NSGRN
      GO TO 100
C-----READ ONE 16 BIT WORD FROM THE DAC BOARD
```

```
    1 CALL BINS(ADAPT,DEVICE,HNDSHK,RAWVAL, STAT)
C-----DECODE BIT 9 WHICH IS THE SIGNAL CONTROLLER STATUS
      NSGRN=BTEST(RAWVAL,9)
  100 CONTINUE
      CALL DGS(NSGRN)
      IC=IC+1
C-----CHECK NUMBER OF PASSES THROUGH MAIN LOOP
C-----IF NUMBER OF PASSES GREATER THAN 100 GET REAL TIME
C-----AND IF MORE THAN ONE OUTPUT INTERVAL HAS ELAPSED
C-----STORE THE RESULTS AND ZERO COUNTERS
        IF(IC.GE.100)THEN
        IC=0
        CALL TTIME
          IF(TIME.GT.TN)THEN
          TN=TN+INTERV
C-----REM IS THE SUBROUTINE THAT STORES COUNTS AND ZEROES COUNTERS
          CALL REM(ICDR)
C-----IF COUNT TIME HAS EXCEEDED THE DESIRED TOTAL TIME STOP
            IF(TN.GT.TNTOT)THEN
            CALL PO
            GO TO 99
            ENDIF
          ENDIF
        ENDIF
C
C ----- MONITOR TRANSITION FROM PHASE TO PHASE
C
      CALL TTIME
      IF(NSGRN.NEQV.NSGRNP)THEN
C
C ----- TRANSITION MADE
C
        TRTF=.TRUE.
        TREND=TIME+TRTIME
        CALL DINCH('TRANSITION',7,NTMDAT+1,68,0)
      ELSE
        IF(TRTF .AND. (TIME.GT.TREND))THEN
C
C ----- TRANSITION TIME IS OVER
C
          TRTF=.FALSE.
          CALL DINCH('          ',7,NTMDAT+1,68,0)
        END IF
      END IF
      NSGRNP=NSGRN
C-----CHECK SIGNAL, TRUE=GREEN OR YELLOW, FALSE=RED
C-----NOTE THAT CONTROLLER IS CONNECTED TO PIN 9 ON
C-----DAC BOARD AND GREEN ON NORTH-SOUTH STREET
C-----MUST PRODUCE CLOSED SWITCH FOR TRUE
        IF(NSGRN)THEN
C-----G1 IS THE PROCESSING ROUTINE FOR GREEN ON NORTH-SOUTH STREET
        CALL G1(ICDR)
```

```
         GO TO 1
         ELSE
C-----G2 IS THE PROCESSING ROUTINE FOR GREEN ON EAST-WEST STREET
         CALL G2(ICDR)
         GO TO 1
         ENDIF
   99 CONTINUE
      END
      BLOCK DATA
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
       INTEGER ST,RT,TM
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      COMMON / DAC / ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      INTEGER*2 ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      COMMON /B/ RES(9),RES1(9)
      LOGICAL RES,RES1
      DATA (RES1(I),I=1,9)/9*.FALSE./,
     *ST/4*0/,LT/4*0/RT/4*0/,J/0/,RAWVAL/0/
      DATA ADAPT/0/,DEVICE/8/,STAT/0/,HNDSHK/0/
      DATA NIO,NTMRC/10,-1/
      DATA ((IDET(I,K),K=1,2),I=1,8)/1,5,0,0,3,8,0,0,0,0,6,4,7,2,0,0/,
     1     NUM/8*0/SUM/8*0./,SUMSQ/8*0./,W1/8*1./,W2/8*19./
      END
      SUBROUTINE G1(ICDR)
C
C    ICDR - ROW FOR DISPLAY OF COUNTS
C
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
       INTEGER ST,RT,TM
      COMMON /B/ RES(9),RES1(9)
      EQUIVALENCE(RES1(9),TRTF)
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      COMMON / DAC / ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      INTEGER*2 ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      LOGICAL RES,RES1,REST,TFTIME,BR(8),TRTF
      DATA (BR(I),I=1,8) /8*.FALSE./
C-----DECODE BITS 1 THROUGH 8 TRYING TO FIND TRUE ON ANY DETECTOR.
C-----IF TRUE IS FOUND, THE ROUTINE CHECKS TO SEE THAT IT IS A CHANGE
C-----FROM THE LAST PASS, IF IT IS NOT A CHANGE, IT IS PROBABLY
C-----THE SAME VEHICLE WHEEL THAT WAS DETECTED IN THE LAST PASS,SO
C-----DON'T PROCESS IT.  THAT IS, WE PROCESS ONLY THE FIRST
C-----DETECTION OF ANY WHEEL
         TFTIME=.FALSE.
         DO 100 I=1,8
```

```
        IF(I.NE.1)GO TO 60
        IF(.NOT.BR(1))GO TO 80
        CALL TTIME
        TFTIME=.TRUE.
        IF(TIME.GT.WIN1(2))THEN
C
C ----- WINDOW HAS PASSED WITHOUT DETECTING RIGHT TURN, CANCEL WINDOW
C
          CALL TMNUMIO(ITMROW(1),ITMCOL(1),1,1,NIO)
          BR(1)=.FALSE.
          GO TO 80
        END IF
   60   CONTINUE
        IF(I.NE.3)GO TO 80
        IF(.NOT.BR(3))GO TO 80
        IF(.NOT.TFTIME)THEN
          CALL TTIME
          TFTIME=.TRUE.
        END IF
        IF(TIME.GT.WIN3(2))THEN
C
C ----- WINDOW HAS PASSED WITHOUT DETECTING RIGHT TURN, CANCEL WINDOW
C
          CALL TMNUMIO(ITMROW(3),ITMCOL(3),3,1,NIO)
          BR(3)=.FALSE.
        END IF
   80   CONTINUE
        REST=BTEST(RAWVAL,I)
        IF(REST.NEQV.RES1(I))THEN
          RES(I)=REST
          RES1(I)=REST
C-----IF BIT IS TRUE AND IS A CHANGE, GO THE PROPER SEGMENT
C-----OF THE PROGRAM AND PROCESS THE DETECTION. EACH SEGMENT HAS
C-----THE NUMBER OF DETECTOR TO WHICH IT RESPONDS
          IF(REST)GO TO (1,2,3,4,5,6,7,8)I
        END IF
        GO TO 100
1       CONTINUE
        IF(BR(1))THEN
C
C ----- RIGHT TURN SEQUENCE IS ACTIVE, EXTEND WINDOW
C
          IF(.NOT.TFTIME)THEN
            CALL TTIME
            TFTIME=.TRUE.
          END IF
          WIN1(2)=TIME+W2(1)
          GO TO 100
        END IF
C
C ----- INITIATE RIGHT TURN SEQUENCE
C
```

```
          IF(.NOT.TFTIME)THEN
            CALL TTIME
            TFTIME=.TRUE.
          END IF
          CALL TMNUMIO(ITMROW(1),ITMCOL(1),1,2,NIO)
          BR(1)=.TRUE.
          WIN1(1)=TIME+W1(1)
          WIN1(2)=TIME+W2(1)
          GO TO 100
2         CONTINUE
          CALL TMNUMIO(ITMROW(2),ITMCOL(2),2,2,NIO)
          IF(TRTF)THEN
            LT(2)=LT(2)+1
            CALL DCC(ICDR,20,LT(2))
          ELSE
            ST(3)=ST(3)+1
            CALL DCC(ICDR,46,ST(3))
          END IF
          CALL TMNUMIO(ITMROW(2),ITMCOL(2),2,1,NIO)
          GO TO 100
3         CONTINUE
          IF(BR(3))THEN
C
C ----- RIGHT TURN SEQUENCE IS ACTIVE, EXTEND WINDOW
C
            IF(.NOT.TFTIME)THEN
              CALL TTIME
              TFTIME=.TRUE.
            END IF
            GO TO 100
          END IF
C
C ----- INITIATE RIGHT TURN SEQUENCE
C
          IF(.NOT.TFTIME)THEN
            CALL TTIME
            TFTIME=.TRUE.
          END IF
          WIN3(2)=TIME+W2(3)
          CALL TMNUMIO(ITMROW(3),ITMCOL(3),3,2,NIO)
          BR(3)=.TRUE.
          WIN3(1)=TIME+W1(3)
          WIN3(2)=TIME+W2(3)
          GO TO 100
4         CONTINUE
          CALL TMNUMIO(ITMROW(4),ITMCOL(4),4,2,NIO)
          IF(TRTF)THEN
            LT(4)=LT(4)+1
            CALL DCC(ICDR,60,LT(4))
          ELSE
            ST(1)=ST(1)+1
            CALL DCC(ICDR,6,ST(1))
```

```
        END IF
        CALL TMNUMIO(ITMROW(4),ITMCOL(4),4,1,NIO)
        GO TO 100
5       CONTINUE
        IF(BR(1))THEN
C
C ----- RIGHT TURN SEQUENCE IS ACTIVE
C
          IF(TIME.LT.WIN1(2))THEN
            IF(TIME.GT.WIN1(1))THEN
C
C ----- DETECTION IS WITHIN WINDOW
C
              IF(.NOT.TFTIME)THEN
                CALL TTIME
                TFTIME=.TRUE.
              END IF
            CALL TMNUMIO(ITMROW(5),ITMCOL(5),5,2,NIO)
            RT(1)=RT(1)+1
            CALL DCC(ICDR,12,RT(1))
            CALL TMNUMIO(ITMROW(5),ITMCOL(5),5,1,NIO)
            ENDIF
          ENDIF
        ELSE
C
C ----- RIGHT TURN SEQUENCE IS NOT ACTIVE
C
          CALL TMNUMIO(ITMROW(5),ITMCOL(5),5,2,NIO)
          LT(3)=LT(3)+1
          CALL DCC(ICDR,40,LT(3))
          CALL TMNUMIO(ITMROW(5),ITMCOL(5),5,1,NIO)
        END IF
        GO TO 100
6       CONTINUE
        CALL TMNUMIO(ITMROW(6),ITMCOL(6),6,2,NIO)
        RT(2)=RT(2)+1
        CALL DCC(ICDR,32,LT(3))
        ST(1)=ST(1)-1
        CALL DCC(ICDR,6,ST(1))
        CALL TMNUMIO(ITMROW(6),ITMCOL(6),6,1,NIO)
        GO TO 100
7       CONTINUE
        CALL TMNUMIO(ITMROW(7),ITMCOL(7),7,2,NIO)
        RT(4)=RT(4)+1
        CALL DCC(ICDR,72,RT(4))
        ST(3)=ST(3)-1
        CALL DCC(ICDR,46,ST(3))
        CALL TMNUMIO(ITMROW(7),ITMCOL(7),7,1,NIO)
        GO TO 100
8       CONTINUE
        IF(BR(3))THEN
C
```

```
C ----- RIGHT TURN SEQUENCE IS ACTIVE
C
          IF(TIME.LT.WIN3(2))THEN
            IF(TIME.GT.WIN3(1))THEN
C
C ----- DETECTION IS WITHIN WINDOW
C
              IF(.NOT.TFTIME)THEN
                CALL TTIME
                TFTIME=.TRUE.
              END IF
             CALL TMNUMIO(ITMROW(8),ITMCOL(8),8,2,NIO)
             RT(3)=RT(3)+1
             CALL DCC(ICDR,52,RT(3))
             CALL TMNUMIO(ITMROW(8),ITMCOL(8),8,1,NIO)
             GO TO 100
             ENDIF
           ENDIF
         ELSE
C
C ----- RIGHT TURN SEQUENCE IS NOT ACTIVE
C
           CALL TMNUMIO(ITMROW(8),ITMCOL(8),8,2,NIO)
           LT(1)=LT(1)+1
           CALL DCC(ICDR,0,LT(1))
           CALL TMNUMIO(ITMROW(8),ITMCOL(8),8,1,NIO)
         END IF
100      CONTINUE
       RETURN
       END
       SUBROUTINE G2(ICDR)
C
C    ICDR - ROW FOR DISPLAY OF COUNTS FOR CURRENT PERIOD
C
       COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
      1        ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
      2        W1(8),W2(8),TIMEE(100,8),IDET(8,2),
      3        RT(4),TM(50,12),ITIM(50,3),J,
      4        NIO,INTERV,INTTOT
        INTEGER ST,RT,TM
       COMMON /B/ RES(9),RES1(9)
       EQUIVALENCE(RES1(9),TRTF)
       COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
       COMMON / DAC / ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
       INTEGER*2 ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
       LOGICAL RES,RES1,REST,TFTIME,BR(8),TRTF
       DATA (BR(I),I=1,8) /8*.FALSE./
C-----DECODE BITS 1 THROUGH 8 TRYING TO FIND TRUE ON ANY DETECTOR.
C-----IF TRUE IS FOUND, THE ROUTINE CHECKS TO SEE THAT IT IS A CHANGE
C-----FROM THE LAST PASS, IF IT IS NOT A CHANGE, IT IS PROBABLY
C-----THE SAME VEHICLE WHEEL THAT WAS DETECTED IN THE LAST PASS,SO
C-----DON'T PROCESS IT.  THAT IS, WE PROCESS ONLY THE FIRST
```

```
C-----DETECTION OF ANY WHEEL
        TFTIME=.FALSE.
        DO 100 I=1,8
        IF(I.NE.6)GO TO 60
        IF(.NOT.BR(6))GO TO 60
        CALL TTIME
        TFTIME=.TRUE.
        IF(TIME.GT.WIN6(2))THEN
C
C ----- WINDOW HAS PASSED WITHOUT DETECTING RIGHT TURN, CANCEL WINDOW
C
          CALL TMNUMIO(ITMROW(6),ITMCOL(6),6,1,NIO)
          BR(6)=.FALSE.
          GO TO 80
        END IF
   60   CONTINUE
        IF(I.NE.7)GO TO 80
        IF(.NOT.BR(7))GO TO 80
        IF(.NOT.TFTIME)THEN
          CALL TTIME
          TFTIME=.TRUE.
        END IF
        IF(TIME.GT.WIN7(2))THEN
C
C ----- WINDOW HAS PASSED WITHOUT DETECTING RIGHT TURN, CANCEL WINDOW
C
          CALL TMNUMIO(ITMROW(7),ITMCOL(7),7,1,NIO)
          BR(7)=.FALSE.
        END IF
   80   CONTINUE
        REST=BTEST(RAWVAL,I)
        IF(REST.NEQV.RES1(I))THEN
          RES(I)=REST
          RES1(I)=REST
C-----IF BIT IS TRUE AND IS A CHANGE, GO THE PROPER SEGMENT
C-----OF THE PROGRAM AND PROCESS THE DETECTION. EACH SEGMENT HAS
C-----THE NUMBER OF DETECTOR TO WHICH IT RESPONDS
          IF(REST)GO TO (1,2,3,4,5,6,7,8)I
        END IF
        GO TO 100
7       CONTINUE
        IF(BR(7))THEN
C
C ----- RIGHT TURN SEQUENCE IS ACTIVE, EXTEND WINDOW
C
          IF(.NOT.TFTIME)THEN
            CALL TTIME
            TFTIME=.TRUE.
          END IF
          WIN7(2)=TIME+W2(7)
          GO TO 100
        END IF
```

```
C
C ----- INITIATE RIGHT TURN SEQUENCE
C
        IF(.NOT.TFTIME)THEN
          CALL TTIME
          TFTIME=.TRUE.
        END IF
        CALL TMNUMIO(ITMROW(7),ITMCOL(7),7,2,NIO)
        BR(7)=.TRUE.
        WIN7(1)=TIME+W1(7)
        WIN7(2)=TIME+W2(7)
        GO TO 100
5       CONTINUE
        CALL TMNUMIO(ITMROW(5),ITMCOL(5),5,2,NIO)
        IF(TRTF)THEN
          LT(3)=LT(3)+1
          CALL DCC(ICDR,40,LT(3))
        ELSE
          ST(4)=ST(4)+1
          CALL DCC(ICDR,66,ST(4))
        END IF
        CALL TMNUMIO(ITMROW(5),ITMCOL(5),5,1,NIO)
        GO TO 100
6       CONTINUE
        IF(BR(6))THEN
C
C ----- RIGHT TURN SEQUENCE IS ACTIVE, EXTEND WINDOW
C
          IF(.NOT.TFTIME)THEN
            CALL TTIME
            TFTIME=.TRUE.
          END IF
          WIN6(2)=TIME+W2(6)
          GO TO 100
        END IF
C
C ----- INITIATE RIGHT TURN SEQUENCE
C
        IF(.NOT.TFTIME)THEN
          CALL TTIME
          TFTIME=.TRUE.
        END IF
        CALL TMNUMIO(ITMROW(6),ITMCOL(6),6,2,NIO)
        BR(6)=.TRUE.
        WIN6(1)=TIME+W1(6)
        WIN6(2)=TIME+W2(6)
        GO TO 100
8       CONTINUE
        CALL TMNUMIO(ITMROW(8),ITMCOL(8),8,2,NIO)
        IF(TRTF)THEN
          LT(1)=LT(1)+1
          CALL DCC(ICDR,0,LT(1))
```

```
        ELSE
          ST(2)=ST(2)+1
          CALL DCC(ICDR,36,ST(2))
        END IF
        CALL TMNUMIO(ITMROW(8),ITMCOL(8),8,1,NIO)
        GO TO 100
2       CONTINUE
        IF(BR(7))THEN
C
C ----- RIGHT TURN SEQUENCE IS ACTIVE
C
          IF(TIME.LT.WIN7(2))THEN
            IF(TIME.GT.WIN7(1))THEN
C
C ----- DETECTION IS WITHIN WINDOW
C
              IF(.NOT.TFTIME)THEN
                CALL TTIME
                TFTIME=.TRUE.
              END IF
            CALL TMNUMIO(ITMROW(2),ITMCOL(2),2,2,NIO)
            RT(4)=RT(4)+1
            CALL DCC(ICDR,72,RT(4))
            CALL TMNUMIO(ITMROW(2),ITMCOL(2),2,1,NIO)
            ENDIF
          ENDIF
        ELSE
C
C ----- RIGHT TURN SEQUENCE IS NOT ACTIVE
C
          CALL TMNUMIO(ITMROW(2),ITMCOL(2),2,2,NIO)
          LT(2)=LT(2)+1
          CALL DCC(ICDR,20,LT(2))
          CALL TMNUMIO(ITMROW(2),ITMCOL(2),2,1,NIO)
        END IF
        GO TO 100
1       CONTINUE
        CALL TMNUMIO(ITMROW(1),ITMCOL(1),1,2,NIO)
        RT(1)=RT(1)+1
        CALL DCC(ICDR,12,RT(1))
        ST(4)=ST(4)-1
        CALL DCC(ICDR,66,ST(4))
        CALL TMNUMIO(ITMROW(1),ITMCOL(1),1,1,NIO)
        GO TO 100
3       CONTINUE
        CALL TMNUMIO(ITMROW(3),ITMCOL(3),3,2,NIO)
        RT(4)=RT(4)+1
        CALL DCC(ICDR,72,RT(4))
        ST(2)=ST(2)-1
        CALL DCC(ICDR,26,ST(2))
        CALL TMNUMIO(ITMROW(3),ITMCOL(3),3,1,NIO)
        GO TO 100
```

```
4       CONTINUE
        IF(BR(6))THEN
C
C ----- RIGHT TURN SEQUENCE IS ACTIVE
C
          IF(TIME.LT.WIN6(2))THEN
            IF(TIME.GT.WIN6(1))THEN
C
C ----- DETECTION IS WITHIN WINDOW
C
              IF(.NOT.TFTIME)THEN
                CALL TTIME
                TFTIME=.TRUE.
              END IF
            CALL TMNUMIO(ITMROW(4),ITMCOL(4),4,2,NIO)
            RT(2)=RT(2)+1
            CALL DCC(ICDR,32,RT(2))
            CALL TMNUMIO(ITMROW(4),ITMCOL(4),4,1,NIO)
            GO TO 100
            ENDIF
          ENDIF
        ELSE
C
C ----- RIGHT TURN SEQUENCE IS NOT ACTIVE
C
          CALL TMNUMIO(ITMROW(4),ITMCOL(4),4,2,NIO)
          LT(4)=LT(4)+1
          CALL DCC(ICDR,60,LT(4))
          CALL TMNUMIO(ITMROW(4),ITMCOL(4),4,1,NIO)
        END IF
100     CONTINUE
      RETURN
      END
      SUBROUTINE PO
C-----THIS SUBROUTINE PROVIDES PRINTER COUNT STATS
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
       INTEGER ST,RT,TM
       OPEN(1,FILE='COUNTS')
       REWIND (1)
         DO 2 I=1,50
         IF((ITIM(I,1).EQ.0) .AND. (ITIM(I,2).EQ.0))GO TO 3
2        WRITE(1,1)(ITIM(I,J),J=1,3),(TM(I,J),J=1,12)
1     FORMAT(3I3,2X,12I5)
3     CONTINUE
      RETURN
      END
      SUBROUTINE REM(ICDR)
C-----THIS SUBROUTINE STORES COUNT DATA EVERY IPOI AND ZEROES COUNTERS
```

```
C
C    ICDR - ROW FOR DISPLAYING COUNTS FOR THIS PERIOD
C
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1        ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2        W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3        RT(4),TM(50,12),ITIM(50,3),J,
     4        NIO,INTERV,INTTOT
      DIMENSION IT(12)
      INTEGER ST,RT,TM
      INTEGER*2 IH,IM,IS,IHUN
      DATA (IT(I),I=1,12) / 12*0 /
      J=J+1
      CALL GETTIM(IH,IM,IS,IHUN)
      ITIM(J,1)=IH
      ITIM(J,2)=IM
      ITIM(J,3)=IS
      K=1
        DO 1 I=1,4
        K1=K+1
        K2=K+2
        TM(J,K)=LT(I)
        TM(J,K1)=ST(I)
        TM(J,K2)=RT(I)
        IT(K)=IT(K)+LT(I)
        IT(K1)=IT(K1)+ST(I)
        IT(K2)=IT(K2)+RT(I)
        LT(I)=0
        RT(I)=0
        ST(I)=0
        K=K+3
1       CONTINUE
C
C ----- DISPLAY ZERO FOR CURRENT COUNTS
C
      CALL DCCZ(ICDR)
C
C ----- DISPLAY TOTAL COUNTS FOR ALL PREVIOUS PERIODS
C
      ICDP=ICDR+1
      CALL DCC(ICDP,0,IT(1))
      CALL DCC(ICDP,6,IT(2))
      CALL DCC(ICDP,12,IT(3))
      CALL DCC(ICDP,20,IT(4))
      CALL DCC(ICDP,26,IT(5))
      CALL DCC(ICDP,32,IT(6))
      CALL DCC(ICDP,40,IT(7))
      CALL DCC(ICDP,46,IT(8))
      CALL DCC(ICDP,52,IT(9))
      CALL DCC(ICDP,60,IT(10))
      CALL DCC(ICDP,66,IT(11))
      CALL DCC(ICDP,72,IT(12))
```

```
      RETURN
      END
      SUBROUTINE INSET
C-----THIS SUBROUTINE EITHER READS THE FILE OF RIGHT TURN
C-----WINDOW DATA OR PROMPTS THE USER FOR IT
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      CHARACTER*1 ANS,ERRMSG*79
      IT1=NTMDAT+2
      CALL ERNLNS(IT1,5,IT1,NIO)
      WRITE(NIO,1)
    1 FORMAT('Do you wish to use the calibration procedure ',/,
     1       'to establish right turn times ?',/,
     2       'Enter Y or N for YES or NO')
      READ(NIO,3)ANS
      IF((ANS.EQ.'Y') .OR. (ANS.EQ.'y'))THEN
        CALL CALSUB
        CALL SK(1)
        WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+1,';72HIS GREEN'
        GO TO 9900
      END IF
      CALL ERNLNS(IT1,4,IT1,NIO)
      WRITE(NIO,2)
    2 FORMAT('Do you wish to read the values for right turn',/,
     *       'times that were previously established ?')
      READ (NIO,3)ANS
    3 FORMAT(A1)
      IF((ANS.EQ.'Y') .OR. (ANS.EQ.'y'))THEN
        OPEN(1,FILE='TIMES')
        REWIND(1)
        DO 5 I=1,8
    5   READ(1,4) AVG(I),SX(I),W1(I),W2(I)
    4   FORMAT(4F10.3)
      ELSE
        CALL ERNLNS(IT1,5,IT1,NIO)
        WRITE(NIO,6)
    6   FORMAT('Enter your chosen values for the earliest and latest',
     *  /,'arrival times at the respective outbound detectors after',
     *  /,'after the prompted inbound detector.')
        IT2=NTMDAT+6
        DO 8 I=1,8
        IF(IDET(I,1).LE.0)GO TO 8
C
C ------ FLASH THE DETECTOR
C
        CALL TMNUMIO(ITMROW(I),ITMCOL(I),I,3,NIO)
        CALL ERNLNS(IT2,4,IT2,NIO)
```

```
        WRITE(NIO,7)I
    7   FORMAT('Inbound detector number',I2)
 1005   CONTINUE
        READ(NIO,*,ERR=1010,IOSTAT=IERR)W1(I),W2(I)
        IF(W1(I).LT.0.0)THEN
          ERRMSG='WINDOW START MUST BE POSITIVE. PLEASE RE-ENTER '//
     1           'ALL DATA.'
          GO TO 1020
        END IF
        IF(W2(I).LE.W1(I))THEN
          ERRMSG='WINDOW END MUST BE GREATER THAN START. '//
     1           'PLEASE RE-ENTER ALL DATA.'
          GO TO 1020
        END IF
        GO TO 1040
 1010   CONTINUE
        ERRMSG='ERROR IN KEYIN. PLEASE RE-ENTER ALL DATA.'
 1020   CONTINUE
        CALL ERNLNS(NTMDAT+7,2,NTMDAT+8,NIO)
        WRITE(NIO,'(A)')ERRMSG
        WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+7,'H'
        GO TO 1005
 1040   CONTINUE
C
C ----- STOP FLASHING THE DETECTOR
C
        CALL TMNUMIO(ITMROW(I),ITMCOL(I),I,1,NIO)
    8   CONTINUE
        OPEN(1,FILE='TIMES')
        REWIND (1)
        DO 2000 I=1,8
        WRITE(1,'(4F10.3)')AVG(I),SX(I),W1(I),W2(I)
 2000 CONTINUE
      ENDIF
 9900 CONTINUE
      RETURN
      END
      SUBROUTINE TI
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      CHARACTER*79 TMDAT
      COMMON / SKEC / TMDAT(15)
      CHARACTER*79 ERRMSG
      CHARACTER FMT1*6,AFMT*5
      PARAMETER(N1=1)
      IT1=NTMDAT+2
      CALL ERNLNS(IT1,2,IT1,NIO)
```

```
      WRITE(NIO,'(A)')'Please specify the interval between '//
     1     'successive count totals (in minutes).'
      IFROM=1
  450 CONTINUE
      READ(NIO,*,ERR=1010,IOSTAT=IERR)INTERV
      IF(INTERV.LE.0)THEN
        ERRMSG='TIME INTERVAL MUST BE POSITIVE.  PLEASE RE-ENTER DATA.'
        GO TO 1020
      END IF
      CALL ERNLNS(IT1,3,IT1,NIO)
      WRITE(NIO,'(A)')'Please specify the total time for which '//
     1     'counts are to be made.'
      IFROM=2
  490 CONTINUE
      READ(NIO,*,ERR=1010,IOSTAT=IERR)INTTOT
      IF(INTTOT.LT.INTERV)THEN
        ERRMSG='TOTAL TIME MUST NOT BE LESS THAN TIME INTERVAL.'//
     1          ' PLEASE RE-ENTER DATA.'
        GO TO 1020
      END IF
      GO TO 4000
 1010 CONTINUE
      ERRMSG='ERROR IN KEYIN.  PLEASE RE-ENTER DATA.'
 1020 CONTINUE
      CALL ERNLNS(NTMDAT+3,2,NTMDAT+4,NIO)
      WRITE(NIO,'(A)')ERRMSG
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+3,'H'
      GO TO (450,490) IFROM
 4000 CONTINUE
      RETURN
      END
      SUBROUTINE DGS(NSGRN)
C
C ----- THIS SUBROUTINE DISPLAYS GREEN STATUS DATA
C
C   NSGREN - LOGICAL VARIABLE, .TRUE. IF NORTH-SOUTH IS GREEN
C
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(3),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      LOGICAL NSGRN,NSGRNP,FIRST
      CHARACTER NS*20,EW*20,AFMT*3
      PARAMETER(N8=8,N11=11,N12=12,AFMT='(A)')
      DATA FIRST / .TRUE. /
      IF(FIRST)THEN
        WRITE(NS,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+1,';68HN-S IS GREEN'
        EW=NS(:8)//'E-W'//NS(12:)
        CALL SCP(NIO)
```

```
        IF(NSGRN)THEN
          WRITE(NIO,AFMT)NS
          NSGRNP=.TRUE.
        ELSE
          WRITE(NIO,AFMT)EW
          NSGRNP=.FALSE.
        END IF
        FIRST=.FALSE.
        GO TO 9900
      END IF
      IF(NSGRN.AND..NOT.NSGRNP)THEN
        CALL SCP(NIO)
        WRITE(NIO,AFMT)NS(:11)
        NSGRNP=.TRUE.
        GO TO 9900
      END IF
      IF(.NOT.NSGRN.AND.NSGRNP)THEN
        CALL SCP(NIO)
        WRITE(NIO,AFMT)EW(:11)
        NSGRNP=.FALSE.
        GO TO 9900
      END IF
      GO TO 9990
 9900 CONTINUE
      CALL RCP(NIO)
 9990 CONTINUE
      RETURN
      END
      SUBROUTINE CGS
C
C ----- THIS SUBROUTINE DISPLAYS GREEN SIGNAL STATUS & PROMPTS FOR
C ----- CONNECTION CONFIRMATION
C
      LOGICAL BTEST
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      COMMON / DAC / ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      INTEGER*2 ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT-4,
     1     ';29HController must be connected to "DAC" board'
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT-3,
     1     ';29Hso that green on North-South has an open switch.'
      IT1=NTMDAT+2
      CALL ERNLNS(IT1,3,IT1,NIO)
      WRITE(NIO,'(A)')'Press any key to confirm signal connection.'
    1 CALL BINS(ADAPT,DEVICE,HNDSHK,RAWVAL,STAT)
      CALL DGS(BTEST(RAWVAL,9))
```

```
      CALL CHKBD(IKBD)
      IF(IKBD.EQ.0)GO TO 1
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT-4,
     1     ';29H                                                    '
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT-3,
     1     ';29H                                                    '
      RETURN
      END
      SUBROUTINE DCC(IR,IC,I)
C
C ----- THIS SUBROUTINE DISPLAYS THE COUNTS FOR THE CURRENT TIME PERIOD
C
C   IR - ROW FOR DISPLAY
C   IC - COLUMN FOR START OF DISPLAY
C   I - VALUE TO DISPLAY
      CHARACTER NLINE*6
      WRITE(NLINE,'(I6)')I
      CALL DINCH(NLINE,7,IR,IC,0)
      RETURN
      END
      SUBROUTINE DCCZ(ICDR)
C
C ----- THIS SUBROUTINE DISPLAYS ALL CURRENT COUNTS AS ZERO
C
C   ICDR - ROW FOR DISPLAY
C
      CALL DCC(ICDR,0,0)
      CALL DCC(ICDR,6,0)
      CALL DCC(ICDR,12,0)
      CALL DCC(ICDR,20,0)
      CALL DCC(ICDR,26,0)
      CALL DCC(ICDR,32,0)
      CALL DCC(ICDR,40,0)
      CALL DCC(ICDR,46,0)
      CALL DCC(ICDR,52,0)
      CALL DCC(ICDR,60,0)
      CALL DCC(ICDR,66,0)
      CALL DCC(ICDR,72,0)
      RETURN
      END
```

```
      SUBROUTINE CALSUB
      LOGICAL RES,BTEST,ERRMSG,WGSDC
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1        ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2        W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3        RT(4),TM(50,12),ITIM(50,3),J,
     4        NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      CHARACTER*79 TMDAT
      COMMON / SKEC / TMDAT(15)
      COMMON / DAC / ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      INTEGER*2 ADAPT.DEVICE,HNDSHK,RAWVAL,STAT
      DATA NDETT / 0 /,ERRMSG/.FALSE./
      IT1=NTMDAT+2
      IT2=NTMDAT+4
      IT3=NTMDAT+5
    3 CONTINUE
      CALL ERNLNS(IT1,5,IT1,NIO)
      WRITE(NIO,4)
    4 FORMAT('Enter the inbound detector number for the right turn',
     */,'movement you wish to monitor, or to stop, enter 9')
   90 CONTINUE
      READ (NIO,'(I1)',ERR=95)NDET
      GO TO 98
   95 CONTINUE
      IF(ERRMSG)THEN
        CALL ERNLNS(IT3,3,IT3,NIO)
      END IF
      WRITE(NIO,'(A,I2,A)')'ERROR IN KEYIN. PLEASE RE-ENTER DATA.'
      ERRMSG=.TRUE.
      CALL ERNLNS(IT2,1,IT2,NIO)
      GO TO 90
   98 CONTINUE
      IF(NDET.EQ.0)NDET=NDETT
      NDETT=NDET
      IF(ERRMSG)THEN
        CALL ERNLNS(IT3,3,IT3,NIO)
        ERRMSG=.FALSE.
      END IF
    1 FORMAT(I1)
      IF(NDET.GT.8)GO TO 5
      IF(NDET.EQ.0)GO TO 105
      DO 100 I=1,8
C
C ----- IS THIS AN INBOUND DETECTOR ?
C
      IF(IDET(I,1).EQ.NDET)GO TO 110
  100 CONTINUE
  105 CONTINUE
      WRITE(NIO,'(A,I2,A)')'DETECTOR',NDET,' IS NOT AN INBOUND '//
     1                     'DETECTOR. PLEASE RE-ENTER DATA.'
```

```
      ERRMSG=.TRUE.
      CALL ERNLNS(IT2,1,IT2,NIO)
      GO TO 90
  110 CONTINUE
      WGSDC=.FALSE.
      GO TO 200
  150 CONTINUE
      WGSDC=.TRUE.
      CALL DINCH('WAITING FOR GREEN ON THIS APPROACH.',135,IT3,0,0)
  200 CONTINUE
C
C ----- IS SIGNAL FOR THIS INBOUND APPROACH GREEN ? IF NOT, WAIT 'TIL IT IS.
C
      CALL BINS(ADAPT,DEVICE,HNDSHK,RAWVAL,STAT)
      CALL DGS(BTEST(RAWVAL,9))
      CALL DDS(RAWVAL)
      IF((NDET.EQ.1) .OR. (NDET.EQ.3))THEN
        IF(.NOT.BTEST(RAWVAL,9))THEN
          IF(WGSDC)GO TO 200
          GO TO 150
        END IF
      ELSE
        IF(BTEST(RAWVAL,9))THEN
          IF(WGSDC)GO TO 200
          GO TO 150
        END IF
      END IF
      IF(WGSDC)THEN
        CALL DINCH('                                          ',7,IT3,0,0)
        WGSDC=.FALSE.
      END IF
  300 CONTINUE
C
C ----- WAIT FOR INBOUND DETECTOR TO CLEAR
C
      CALL BINS(ADAPT,DEVICE,HNDSHK,RAWVAL,STAT)
      CALL DGS(BTEST(RAWVAL,9))
      CALL DDS(RAWVAL)
      IF(BTEST(RAWVAL,NDET))THEN
        IF(WGSDC)GO TO 300
        WGSDC=.TRUE.
        CALL DINCH('WAITING FOR INBOUND DETECTOR TO CLEAR.',135,IT3,0,0)
        GO TO 300
      END IF
      IF(WGSDC)THEN
        CALL DINCH('                                            ',7,IT3,0,0)
      END IF
C
C ----- FLASH THE SELECTED DETECTOR
C
      CALL TMNUMIO(ITMROW(NDET),ITMCOL(NDET),NDET,3,NIO)
      J=1
```

```
C-----GET ONE 16 BIT WORD FROM DAC BOARD
    2 CALL BINS(ADAPT,DEVICE, HNDSHK,RAWVAL,STAT)
      RES=BTEST(RAWVAL,9)
      CALL DGS(RES)
      IF((NDET.EQ.1) .OR. (NDET.EQ.3))THEN
        IF(.NOT.RES)THEN
          CALL TMNUMIO(ITMROW(NDET),ITMCOL(NDET),NDET,1,NIO)
          GO TO 150
        END IF
      ELSE
        IF(RES)THEN
          CALL TMNUMIO(ITMROW(NDET),ITMCOL(NDET),NDET,1,NIO)
          GO TO 150
        END IF
      END IF
C-----DECODE THE BIT THAT CORRESPONDS TO THE DETECTOR BEING STUDIED
      RES=BTEST(RAWVAL,IDET(NDET,J))
        IF(RES)THEN
C
C ----- BRIGHTEN THE SELECTED DETECTOR THAT HAS BEEN TRIPPED
C
        IT4=IDET(NDET,J)
        CALL TMNUMIO(ITMROW(IT4),ITMCOL(IT4),IT4,2,NIO)
        CALL TTIME
        TIMEX(J)=TIME
        J=J+1
          IF(J.LE.2)THEN
          GO TO 2
          ELSE
C
C ----- WAIT FOR OUTBOUND DETECTOR TO CLEAR
C
   50   CONTINUE
        CALL BINS(ADAPT,DEVICE,HNDSHK,RAWVAL,STAT)
        IF(BTEST(RAWVAL,IDET(NDET,2)))GO TO 50
C
C ----- DIM BOTH DETECTORS
C
        CALL TMNUMIO(ITMROW(IT4),ITMCOL(IT4),IT4,1,NIO)
        IT4=IDET(NDET,1)
        CALL TMNUMIO(ITMROW(IT4),ITMCOL(IT4),IT4,1,NIO)
C-----CALL THE SUBROUTINE THAT COMPUTES AND STORES TIME DATA
          CALL OUT
          GO TO 3
          ENDIF
        ELSE
        GO TO 2
        ENDIF
    5 CONTINUE
C-----CALL THE SUBROUTINE WRITES OUTPUT FILE AND SCREEN STATS
      CALL OUT1
      RETURN
```

```
      END
      SUBROUTINE OUT
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      CHARACTER*1 ANS
C-----PROMPT USER FOR HIS DESIRES TO USE THE CURRENT RECORD
      IT1=NTMDAT+2
      CALL ERNLNS(IT1,5,IT1,NIO)
      TT=TIMEX(2)-TIMEX(1)
      WRITE(NIO,'(A,I2,A,I2,A,F6.2,A)')'Travel time from detector',
     1     IDET(NDET,1),' to detector',IDET(NDET,2),
     2     ' was',TT,' seconds.'
      WRITE(NIO,1) NDET
    1 FORMAT('Do you wish to use this record for detector',I2,' ?')
      READ(NIO,2) ANS
    2 FORMAT(A1)
C-----PROCESS THE RECORD IF THE USER ENTERS ANYTHING BUT N
        IF(ANS.NE.'N')THEN
        NUMT=NUM(NDET)+1
        NUM(NDET)=NUMT
        TIMEE(NUMT,NDET)=TT
        SUMT=SUM(NDET)+TT
        SUM(NDET)=SUMT
        SUMSQ(NDET)=SUMSQ(NDET)+TT*TT
        IF(NUMT.GT.1)THEN
          SX(NDET)=SQRT((SUMSQ(NDET)-(SUMT*SUMT)/NUMT)/(NUMT-1))
        ELSE
          SX(NDET)=0.0
        END IF
        AVG(NDET)=SUMT/NUMT
        ENDIF
      RETURN
      END
      SUBROUTINE OUT1
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      DO 5 I=1,8
      XT1=NUM(I)
      IF(XT1.LE.0.0)GO TO 5
      XT2=SX(I)/SQRT(XT1)
      XT2=XT2+XT2
      XT1=AVG(I)
      W1(I)=XT1-XT2
```

```
      W2(I)=XT1+XT2
    5 CONTINUE
C
C ----- REVISE WINDOWS ?
C
      CALL RWS
      OPEN(1,FILE='TIMES')
      DO 20 I=1,8
C-----WRITE THE OUTPUT FILE
      WRITE(1,1) AVG(I),SX(I),W1(I),W2(I)
   20 CONTINUE
    1 FORMAT(4F10.3)
    4 FORMAT(3X,I2,8X,F6.3,5X,F6.2,7X,F6.2,6X,F6.2)
    3 CONTINUE
      CLOSE(1)
      RETURN
      END
      SUBROUTINE RWS
      LOGICAL ERRMSG
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1        ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2        W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3        RT(4),TM(50,12),ITIM(50,3),J,
     4        NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      DATA ERRMSG/.FALSE./,NDETT/0/
  100 CONTINUE
C
C ----- CLEAR THE SCREEN
C
      WRITE(NIO,'(2A)')CHAR(27),'[2J'
      WRITE(NIO,2)
    2 FORMAT('AVERAGE TURNING MOVEMENT TIMES BY ',
     *'INBOUND DETECTOR NUMBER:',//,'DETECTOR  AVERAGE TIME',
     *'  STD DEV  WINDOW START  WINDOW END')
      II=5
      DO 1000 I=1,8
      IF(IDET(I,1).EQ.0)GO TO 1000
      II=II+1
C-----WRITE THE STATS TO THE SCREEN
      WRITE(NIO,4)I,AVG(I),SX(I),W1(I),W2(I)
    4 FORMAT(3X,I2,8X,F6.3,5X,F6.2,7X,F6.2,6X,F6.2)
 1000 CONTINUE
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',II,'H'
      WRITE(NIO,'(A/A)')'Enter an inbound detector number '//
     1  'for which','you wish to revise the window, or to stop, '//
     2  'enter 9.'
 1900 CONTINUE
      READ(NIO,'(I1)',ERR=2000)NDET
      IF(NDET.EQ.0)THEN
        NDET=NDETT
      ELSE
```

```
        NDETT=NDET
      END IF
      GO TO 2010
 2000 CONTINUE
      IF(ERRMSG)THEN
        IT3=II+3
        CALL ERNLNS(IT3,3,IT3,NIO)
      END IF
      WRITE(NIO,'(A,I2,A)')'ERROR IN KEYIN. PLEASE RE-ENTER DATA.'
      ERRMSG=.TRUE.
      IT2=II+2
      CALL ERNLNS(IT2,1,IT2,NIO)
      GO TO 1900
 2010 CONTINUE
      IF(NDET.EQ.0)GO TO 3010
      IF(NDET.EQ.9)GO TO 9990
      DO 3000 I=1,8
C
C ----- IS THIS AN INBOUND DETECTOR ?
C
      IF(IDET(I,1).EQ.NDET)GO TO 3020
 3000 CONTINUE
 3010 CONTINUE
      WRITE(NIO,'(A,I2,A)')'DETECTOR',NDET,' IS NOT AN INBOUND '//
     1                     'DETECTOR. PLEASE RE-ENTER DATA.'
      ERRMSG=.TRUE.
      IT2=II+2
      CALL ERNLNS(IT2,1,IT2,NIO)
      GO TO 1900
 3020 CONTINUE
      CALL SC(NDET)
      CALL STMODE(2)
      GO TO 100
 9990 CONTINUE
      RETURN
      END
      SUBROUTINE TTIME
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      INTEGER*2 IH,IM,IS,IHUN
      CALL GETTIM(IH,IM,IS,IHUN)
      ITIME=IS+IM*60+IH*3600
      TIME=IHUN
      TIME=TIME*.01
      TIME=TIME+ITIME
      RETURN
      END
      SUBROUTINE SK(IA)
```

```
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),SX(8),
     1        ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2        W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3        RT(4),TM(50,12),ITIM(50,3),J,
     4        NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      CHARACTER*79 TMDAT
      COMMON / SKEC / TMDAT(15)
      CHARACTER FMT1*6,AFMT*5,I2FMT*6
      PARAMETER(N1=1,FMT1='(20I4)',AFMT='(A79)',I2FMT='(I2.2)')
      IF(NTMRC.GE.0)GO TO 210
C
C ----- FILE WITH SKETCH DATA
C
      OPEN(4,FILE='TM.DAT')
      REWIND 4
C
C ----- READ NUMBER OF DETECTORS IN SKETCH
C
      READ(4,FMT1)NTMRC
C
C ----- READ ROW NUMBER FOR EACH DETECTOR ID IN THE SKETCH
C
      READ(4,FMT1)itmrow(1),itmrow(2),itmrow(3),itmrow(4),itmrow(5),
     1            itmrow(6),itmrow(7),itmrow(8)
C
C ----- READ COLUMN NUMBER FOR EACH DETECTOR ID IN THE SKETCH
      READ(4,FMT1)itmcol(1),itmcol(2),itmcol(3),itmcol(4),itmcol(5),
     1            itmcol(6),itmcol(7),itmcol(8)
      NTMDAT=N1
      DO 50 I=N1,NTMRC
      ITMCOL(I)=ITMCOL(I)-N1
   50 CONTINUE
  100 CONTINUE
C
C ----- READ DATA FOR SKETCH
C
      READ(4,AFMT,END=200)TMDAT(NTMDAT)
      NTMDAT=NTMDAT+N1
      GO TO 100
  200 CONTINUE
      NTMDAT=NTMDAT-N1
  210 CONTINUE
C
C ----- CLEAR SCREEN
C
      WRITE(NIO,'(2A)')CHAR(27),'[2J'
C
C ----- WRITE SKETCH TO TERMINAL
C
      DO 250 I=N1,NTMDAT
```

```
      WRITE(NIO,AFMT)TMDAT(I)
  350 CONTINUE
      CALL SCP(NIO)
      DO 300 I=N1,NTMRC
C
C ----- PUT DETECTOR ID NUMBERS ON SKETCH
C
      CALL NTD(I,1,ITMROW(I),ITMCOL(I))
  300 CONTINUE
      CALL RCP(NIO)
      RETURN
      END
      SUBROUTINE TMNUMIO(IROW,ICOL,INUM,I,NIO)
C
C ----- THIS SUBROUTINE WRITES A SINGLE NUMBER TO A SPECIFIC ROW AND COLUMN
C ----- THE NUMBER CAN BE NORMAL INTENSITY, BRIGHT OR BLINKING
C
C   IROW - INTEGER ROW
C   ICOL - INTEGER COLUMN
C   INUM - NUMBER TO BE WRITTEN
C   I - METHOD OF DISPLAY:
C       1 = NORMAL INTENSITY
C       2 = BRIGHT
C       3 = BLINKING
C   NIO - UNIT NUMBER THAT IS OPENED FOR I/O TO THE TERMINAL
C
      CHARACTER*1 GR(3)
      DATA (GR(I),I=1,3) /'0','1','5'/
      CALL SCP(NIO)
      CALL NTD(INUM,I,IROW,ICOL)
      CALL RCP(NIO)
      RETURN
      END
      SUBROUTINE SCP(NIO)
C
C ----- THIS SUBROUTINE SAVES THE CURRENT CURSOR POSITION (ROW AND COLUMN).
C ----- THE CURSOR CAN BE MOVED TO THE MOST CURRENT SAVED POSITION BY
C ----- CALLING SUBROUTINE RCP.
C   NIO - UNIT NUMBER THAT IS OPEN FOR I/O TO CONSOLE
C
      WRITE(NIO,'(2A)')CHAR(27),'[s'
      RETURN
      END
      SUBROUTINE RCP(NIO)
C
C ----- THIS SUBROUTINE RESTORES THE CURSOR TO THE MOSE CURRENT SAVED POWITION.
C ----- CURSOR POSITION CAN BE SAVED BY CALLING SUBROUTINE SCP.
C   NIO - UNIT NUMBER THAT IS OPEN FOR I/O TO CONSOLE.
C
      WRITE(NIO,'(4A)')CHAR(27),'[u',CHAR(27),'[A'
      RETURN
      END
      SUBROUTINE CDC
```

```
C
C ----- THIS SUBROUTINE DISPLAYS THE DETECTOR SKETCH AND PROMPTS FOR
C ----- CONFIRMATION OF DETECTOR CONNECTION TO DAC
C
      CHARACTER AFMT*3
      LOGICAL BTEST
      COMMON /A/ TIME,NDET,AVG(8),TIMEX(2),NUM(8),SUM(8),SUMSQ(8),S((8),
     1       ST(4),LT(4),WIN1(2),WIN6(2),WIN3(2),WIN7(2),
     2       W1(8),W2(8),TIMEE(100,8),IDET(8,2),
     3       RT(4),TM(50,12),ITIM(50,3),J,
     4       NIO,INTERV,INTTOT
      INTEGER ST,RT,TM
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      COMMON / DAC / ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      INTEGER*2 ADAPT,DEVICE,HNDSHK,RAWVAL,STAT
      PARAMETER(N1=1,N2=2,AFMT='(A)')
C
C ----- DISPLAY SKETCH WITH DETECTOR NUMBERS
C
      CALL SK(N1)
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT-4,
     1      ';29HDetectors must be connected to the "DAC"'
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT-3,
     1      ';29Hboard using the numbering sequence at left.'
C
C ----- ISSUE PROMPT TO LEAVE 1 BLANK LINE BELOW SKETCH
C
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT+N2,'H'
      WRITE(NIO,AFMT)'Press any key to confirm detector connections.'
  100 CONTINUE
      CALL BINS(ADAPT,DEVICE,HNDSHK,RAWVAL,STAT)
      CALL DDS(RAWVAL)
      CALL CHKBD(IKBD)
      IF(IKBD.EQ.0)GO TO 100
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT-4,
     1      ';29H                                                  '
      WRITE(NIO,'(2A,I2.2,A)')CHAR(27),'[',NTMDAT-3,
     1      ';29H                                                  '
      RETURN
      END
      SUBROUTINE ERNLNS(IR1,N,IR,NIO)
C
C ----- THIS SUBROUTINE ERASES LINES ON THE SCREEN & PLACES CURSOR
C ----- "ANSI.SYS" MUST BE INSTALLED
C
C   IR1 - FIRST ROW TO ERASE
C   N - NUMBER OF ROWS TO ERASE
C   IR - ROW WHERE CURSOR IS PLACED AFTER ERASING
C        (LINE 0 IS AT TOP OF SCREEN)
C        (COLUMN 0 IS AT LEFT OF SCREEN)
C   NIO - UNIT WITH FILE OPEN FOR CONSOLE I/O
C
```

```
      CHARACTER*2 CR1,CR,CC,I2FMT*6,C1*5,C2*3,C3*5
      LOGICAL FIRST
      PARAMETER (N1=1,N3=3,N4=4,I2FMT='(I2.2)')
      DATA FIRST / .TRUE. /
      DATA C1 / ' [  4' /
      DATA C2 / ' [K' /
      IF(FIRST)THEN
        C1(:N1)=CHAR(27)
        C2(:N1)=C1(:N1)
        C3=C1
        FIRST=.FALSE.
      END IF
      WRITE(C1(N3:N4),I2FMT)IR1
      WRITE(C3(N3:N4),I2FMT)IR
      WRITE(NIO,'(99(A,:,/))')C1,(C2,I=N1,N),C3
      RETURN
      END
      SUBROUTINE DDS(RAWVAL)
C
C ----- THIS SUBROUTINE DISPLAYS THE CURRENT DETECTOR STATUS
C
C   RAWVAL - EACH BIT REPRESENTS THE CONDITION OF A DETECTOR:
C            1 - DETECTOR IS TRIPPED
C            0 -              NOT TRIPPED
C            BIT 1 IS FOR DETECTOR 1, ETC.
C
      INTEGER*2 RAWVAL
      LOGICAL BTEST,BR(8)
      PARAMETER(N1=1)
      COMMON / SKE / NTMRC,ITMROW(20),ITMCOL(20),NTMDAT
      DATA (BR(I),I=1,8)/8*.FALSE./
      DO 200 I=N1,NTMRC
      IF(BTEST(RAWVAL,I))THEN
C
C ----- DETECTOR IS TRIPPED
C
        IF(.NOT.BR(I))THEN
C
C ----- DETECTOR WAS NOT PREVIOUSLY TRIPPED, BRIGHTEN DETECTOR
C
          CALL TMNUMIO(ITMROW(I),ITMCOL(I),I,2,NIO)
          BR(I)=.TRUE.
        END IF
      ELSE
C
C ----- DETECTOR IS NOT TRIPPED
C
        IF(BR(I))THEN
C
C ----- DETECTOR WAS PREVIOUSLY TRIPPED
C
          CALL TMNUMIO(ITMROW(I),ITMCOL(I),I,N1,NIO)
```

```
          BR(I)=.FALSE.
        END IF
      END IF
  200 CONTINUE
      RETURN
      END
      SUBROUTINE NTD(N,I,IR,IC)
C
C ----- THIS SUBROUTINE DISPLAYS A SINGLE DIGIT NUMBER AT A SPECIFIED
C ----- ROW AND COLUMN, ON PAGE 0
C
C   N - NUMBER TO DISPLAY
C   I - METHOD OF DISPLAY:
C       1 - NORMAL INTENSITY
C       2 - BRIGHT
C       3 - BLINKING
C   IR - ROW
C   IC - COLUMN
C
      INTEGER GR(3)
      DATA (GR(I),I=1,3) / 7,15,135 /
      CALL DINCH(CHAR(N+ICHAR('0')),GR(I),IR,IC,0)
      RETURN
      END
```