TECHNICAL MEMORANDUM (NASA) 85

ORIGINAL PAGE IS
OF POOR QUALITY

A MICROCOMPUTER-BASED POSITION UPDATING SYSTEM FOR GENERAL AVIATION UTILIZING LORAN-C

This report identifies the advantages Loran-C offers over VOR/DME as a primary navigation aid. These advantages include long range coverage by several stations; coverage at low altitudes and the capability to have non-precision approaches at airports not already served by landing aids. Modern digital electronic technology is used to produce a device to convert Loran-C to useful pilot information using simple software algorithms and low-cost microprocessor devices. The cost and lack of availability of suitable processors to execute these algorithms have prevented a Loran-C navigator from being developed prior to this work. Results indicate that the microprocessor-based Loran-C navigator has an accuracy of 1.0 nm or less over an area typically covered by a triad of Loran-C stations and can execute a position update in less than 0.2 seconds. The system has been tested in 30 hours of flight and has proved that it can give reliable and accurate navigation information.

by

Joseph P. Fischer

Avionics Engineering Center
Department of Electrical Engineering
Ohio University
Athens, Ohio 45701

May 1982

Prepared for

NASA-Langley Research Center
Hampton, Virginia 23665

Grant NGR 36-009-017

(NASA-CR-169722) A MICROCOMPUTER-BASED POSITION UPDATING SYSTEM FOR GENERAL AVIATION UTILIZING LORAN-C (Ohio Univ.) 121 p HC A06/MF A01 CSCL 17G

N83-15277

Unclas
G3/04 08098

TABLE OF CONTENTS

ORIGINAL PAGE IS
OF POOR QUALITY

LIST OF FIGURES

PAGE

ORIGINAL PAGE IS
OF POOR QUALITY

LIST OF FIGURES (CONTINUED)

LIST OF FIGURES (CONTINUED)

LIST OF TABLES

ORIGINAL PAGE IS
OF POOR QUALITY

I. INTRODUCTION AND STATEMENT OF PROBLEM

Loran-C, a 100 KHz, time-difference, hyperbolic radio navigation scheme, has certain advantages for use as a primary navigation aid for general-aviation aircraft. Significant advantages include long-range coverage by using a few transmitters, useful signals at low altitudes and in mountainous terrain, and signals which are stable allowing accurate and consistent positional data to be obtained. These advantages have prompted federal authorities to consider this system as a primary navigation aid and for application as a non-precision (two-dimensional) landing aid.[1]

Loran-A, which operates at 2.5 MHz, was in use for over 30 years but has now been discontinued. Loran-C, which has been in operation for 20 years, is continuing to expand with new chains in all parts of the world. Loran-C provides more stable signals than Loran-A over long distances and for this reason, there has been increased interest in land-based use of Loran-C for aviation activities. Loran has traditionally been used by the maritime services and has not been accepted previously in aviation circles because of users' unfamiliarity with hyperbolic systems and their impracticality and the lack of low-cost, simple equipment to convert the hyperbolic data to a more readily understood form.

This paper documents a rather complete effort to design, build, and flight test a navigator unit which converts Loran-C data at the output of a receiver to latitude/longitude coordinates and also range/bearing coordinates to a waypoint. This work has taken advantage of contemporary microprocessor technology for simple implementation of new algorithms to give accurate positional information from Loran-C. This has involved investigation and testing of various data conversion methods, designing and building the necessary microcomputer hardware and the writing of the data conversion software. Finally, the entire concept was tested during approximately 30 hours of flight in a Piper Cherokee and a Douglas DC-3. It was proved that a simple method will indeed convert Loran-C data to a pilot-usable form. The microcomputer equipment necessary is low in cost and readily available and the system works well in providing good navigation information in an aircraft flight environment. With the continuing trend toward more powerful and lower cost electronic devices, the roadblocks to implementing fully a Loran-C navigator have essentially been removed.

The international standard method of navigating is to use the VOR (VHF Omni Range) and DME (Distance Measuring Equipment). This system, with over 1000 stations in the United States, has proven to be acceptable

---

[1]Non-precision is used to connote that no vertical guidance is provided. An extensive study in this area has been ongoing in the state of Vermont. See, for example, Polhemus, W.L., "Evaluation of Loran-C for Enroute Navigation and non-Precision Approach Within the State of Vermont," Proceedings of the National Aerospace Symposium, Dayton, Ohio, March 1980.

and very capable. Information is provided to the pilot in the form of the angular bearing and distance relative to a VOR/DME station [1]. A more recent feature called area navigation (RNAV) works in conjunction with VOR/DME data to allow the pilot to set up arbitrary waypoints (at airports, VOR/DME stations, etc.) and then fly a straight-line course between these points.

The incentive to consider low-frequency systems is that the VHF and UHF signals from a VOR/DME station do not propagate to follow the earth and provide low altitude coverage and signals and can be blocked by mountains. Also, with low-frequency systems, 20 to 30 stations could cover the entire continental U.S. which would be considerably cheaper and easier to maintain than over 1000 stations.

With the ability to cover low altitudes, Loran-C could provide, at no extra cost, an accurate approach navigation system currently unavailable at many airports. There are over 14,000 airports in the U.S., half of which do not have electronic instrument landing aids. By having an all-weather approach aid, such as Loran-C could provide, the utility and cost effectiveness of existing airports could be increased.

Clearly, there are other methods of air navigation including non-directional beacons (NDB), dead reckoning or inertial navigation. There is also some promise of using satellites for navigation [2]. These methods have disadvantages. For example, the NDB is not as precise to use as the Loran-C and inertial navigation and satellite systems are certainly much more expensive than Loran-C.

The present-day, Loran-C network currently covers much of the northern hemisphere as shown in Figure 1-1. The possibility of using this system especially for non-precision approaches and RNAV has generated much interest and has been investigated extensively [3,4]. This research has dealt with ways to handle such practical problems as GDOP (geometric dilution of precision), which occurs when the hyperbolic lines-of-position cross at acute angles, and the effect of passing over a Loran-C station. It has been shown that these problems can be handled effectively without seriously affecting the overall utility of the system [5,6].

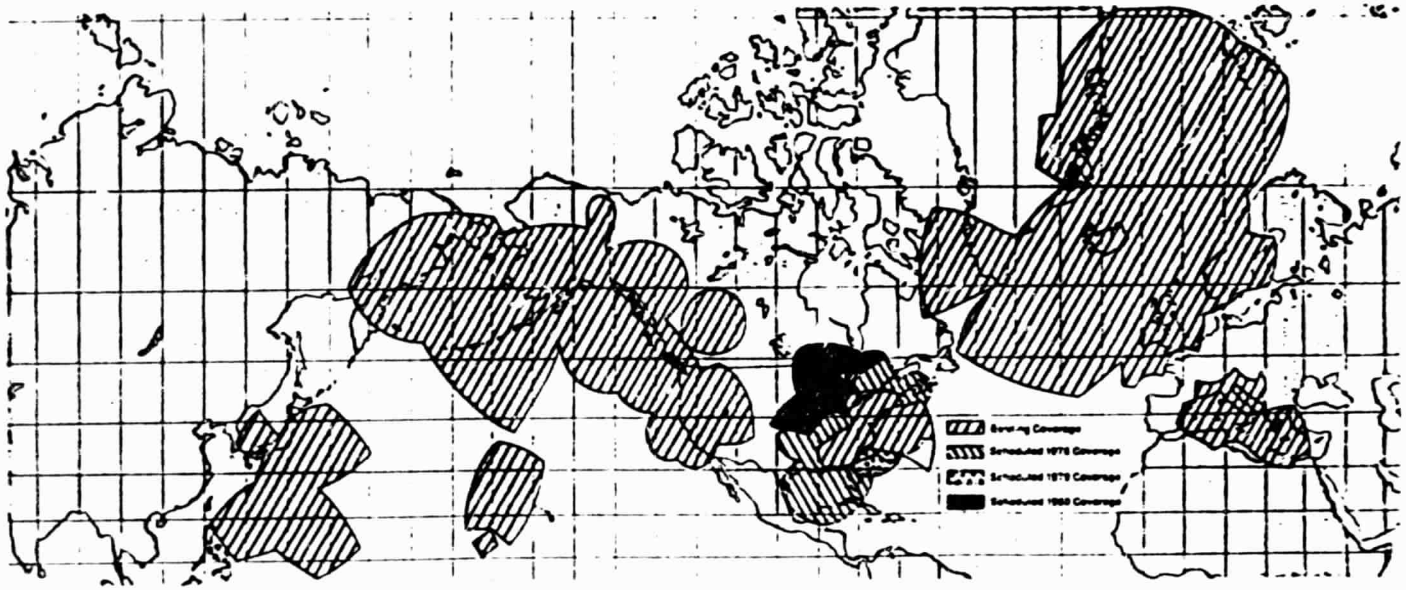

Figure 1-1. Present Loran-C Coverage is in the Northern Hemisphere and is Shown Here. Source: U. S. Coast Guard.

II. REVIEW OF LORAN-C DEVELOPMENT AND OPERATION

ORIGINAL PAGE IS
OF POOR QUALITY

This brief review of low-frequency navigation systems, including Loran, is intended to show how present-day Loran-C has benefitted from experiences gained through experiments with other systems. The operation of the Loran-C system is discussed, and the details of how navigation information is derived from Loran-C are presented.

A. Historical Development. Early in aviation history it became apparent that flying activities needed to be conducted at times when navigation by visual cues alone was not adequate. Other means of navigation were needed, and these can be divided into self-contained and ground-based systems. Self-contained systems such as inertial navigation have all the navigation components on board the aircraft. A ground-based system sends a signal or signals to the aircraft where navigation information is derived. Ground-based systems, such as VORs, are preferred because the heavy and bulky equipment can be kept off the aircraft. In addition, use of ground-based navigation systems can provide correlation among all users to a fixed reference point (or a set of points fixed with respect to the users) thus allowing users to establish their position and avoid collisions with each other.

1. Reasons for Low-Frequency Systems. It is most natural to use radio signals to transmit navigation information to an aircraft, although there are restrictions as to which frequencies are optimum. As shown in Table 2-1 the propagation characteristics of various frequency regions vary considerably [7,8]; at very low frequencies the transmitted waveform follows the earth's curvature in a duct formed between the earth and the ionosphere, and in the VHF region the waveform follows a straight-line path from the transmitter to the receiver and can be blocked by the horizon or large objects such as mountains.

While low-frequency signals may be transmitted over several hundreds or thousands of kilometers, with sufficient power, these signals often are contaminated with noise to the extent that the level of average noise is higher than the signal. The transmission of medium-frequency signals for navigation use has a serious disadvantage in that these signals are refracted by the ionosphere causing the path length of the signal to change as the height of the ionosphere varies with time of day and season. The use of VHF or higher frequencies affords bandwidths of several kilohertz to megahertz for transmitting navigation information; however, with line-of-sight coverage, the cost of placing and maintaining the transmitters necessary to provide adequate coverage is a serious disadvantage.

For covering a large area of several hundred thousand kilometers or more, a low-frequency system appears appropriate. At very low frequencies, transmitter and antenna design is difficult because of the high power levels and long wavelengths involved. The bandwidth

ORIGINAL PAGE IS
OF POOR QUALITY

Frequency Range	Wavelength (meters)	Designation	Type of Propagation
3Hz - 30KHz	10^5 to 10^4	very low freq.	Global
30KHz to 300KHz	10^4 to 10^3	low freq.	>2000 Km
300Khz to 3MHz	10^3 to 10^2	medium freq.	ground wave 500 to 1000 Km skywave 1000-2000 Km
3MHz to 30MHz	10^2 to 10^1	high freq.	ground wave limited to <500 Km, skywave can cover several thousand Km
30MHz to 300MHz	10^1 to 10^0	very high freq.	mostly line-of-sight
300MHz to 3GHz and above	10^0 to 10^{-1} 10-1-	ultra high freq.	line-of-sight

Table 2-1. Range of Radio Frequencies and Their Propagation Characteristics.

ORIGINAL PAGE IS
OF POOR QUALITY

available at very low frequencies is narrow, less than that required for voice transmission, thus limiting how much information may be sent [9]. The low-frequency band allows enough bandwidth to use pulse-type navigation systems which simplify detection of navigation information compared to the use of continuous-wave systems.

2. Contributions of Other Systems. Many tests were conducted using low frequencies for navigation. Some of the tests led to systems that did work, although they were not very practical and were difficult to use. Because of the stable nature of very-low-frequency signals, some systems were developed which used variations in phase relative to receiver location. The use of continuously transmitted signals was necessary because of the bandwidth restriction on the antennas. The European Decca and the more recently developed Omega systems are examples of these. In these systems, the receiver compares the relative phase between two or more stations, and these phase differences then define a set of hyperbolic lines of position, the intersection of which is the receiver's location. A lane identification problem is inherent to these systems because of the cyclic redundancy of the transmitted signal. There can be considerable variation of the transmitted phase because of natural phenomena, and this has been the subject of much research [10,11].

In the 1940s, work was begun on pulse-type navigation systems operating at various frequencies, mostly in the medium-frequency range [12]. These systems attempted to derive navigation information by using the time difference of arrival between pulses received from different stations. Over a period of time, the technology of designing highly stable oscillators became very refined and it was soon recognized that much better navigation coverage could be obtained in the low-frequency band. Because of the demands of the Second World War, a pulse-type system, called Loran-A and operating near 2 MHz, was established while research on low-frequency techniques continued. In 1946, a system called Cyclan using automatic cycle identification and phase measuring techniques was developed. This soon led to the testing of another system called Cytac, operating in the 90-100 kHz band, which used the leading edge of pulses to measure time differences. In the late 1950s, a set of specifications was developed which was considerably improved over the standard Loran, and thus the Cytac tests became what is known today as Loran-C [13]. The Loran-A system was then phased out beginning in the late 1970s with complete withdrawal of the system in 1980.[2]

Throughout many early experiments there was very little known about the propagation characteristics at low frequencies and how these characteristics would influence the outcome of the navigation experiment. Often, some rather surprising discoveries were made during the

---

[2] A good discussion of these navigation methods is provided by Sandretto, P.C., Aeronautical Radio Engineering, McGraw-Hill Book Company, New York, 1942.

testing of these prospective systems and much knowledge was gained by actual field work. Loran-C today contains many elements such as long-range coverage and pulse transmission to avoid mutual interference and skywave contamination, which were discovered years earlier. The system is an optimization of many of its predecessors.

F. Basics of Loran-C Time Differences (TD). Each Loran-C station transmits a series of pulses with a carrier frequency of 100 kHz. An example of a transmitted pulse is shown in Figure 2-1. The mathematical description of this pulse is:

$$f(t) = \left(\frac{t}{t_p}\right)^2 \exp\left[2\left(t - \frac{t}{t_p}\right)\right] \cos\omega_c t$$

where t_p = 65 μs and ω_c = 2π×100 kHz. This equation represents a pulse with rapid rise time to allow the receiver to lock onto the early part of the pulse before a skywave pulse appears. The latter part of the pulse has a slow decay time to confine the energy to a band extending from 80 to 120 kHz. The phase of the leading edge is tightly controlled since this is where the receiver samples the signal to determine time differences [14].

1. Development of Time Differences. Loran-C stations are associated in groups of between three and five stations; one station is designated the master and the others are designated secondary or slave stations. As shown in Figure 2-2, the master station transmits nine pulses, the first eight separated from each other by 1000 μs, and the ninth separated from the eighth by 2000 μs. This is done for identification purposes. Each of the secondary stations transmits eight pulses separated from each other by 1000 μs. A particular group of stations is called a chain and is identified by the chain's group repetition interval (GRI) which is the rate that a station transmits pulse groups. Further identification is provided by periodically reversing the phase of the carrier for each station in a set sequence [15]. Typical GRIs are in the range of 49900 μs to 99900 μs. By measuring the amount of time between the arrival of a pulse set for different stations, the Loran-C time differences are determined which can then be used to locate the receiver's position. This will be examined in more detail in the next chapter.

2. Hyperbolic Coordinates. As will be pointed out later, the Loran-C TD readings are of the form of a hyperbolic equation. This places Loran-C with most other navigation systems used below 30 MHz whose primary source of information is a set of hyperbolic coordinates. The reason for this is that narrow bandwidths at low frequencies do not allow sufficient data rates to transmit signals with a higher information content than the pulse system. Unfortunately, it also puts more of a burden on the receiver or operator who must then convert the coordinates to a more convenient form. The most

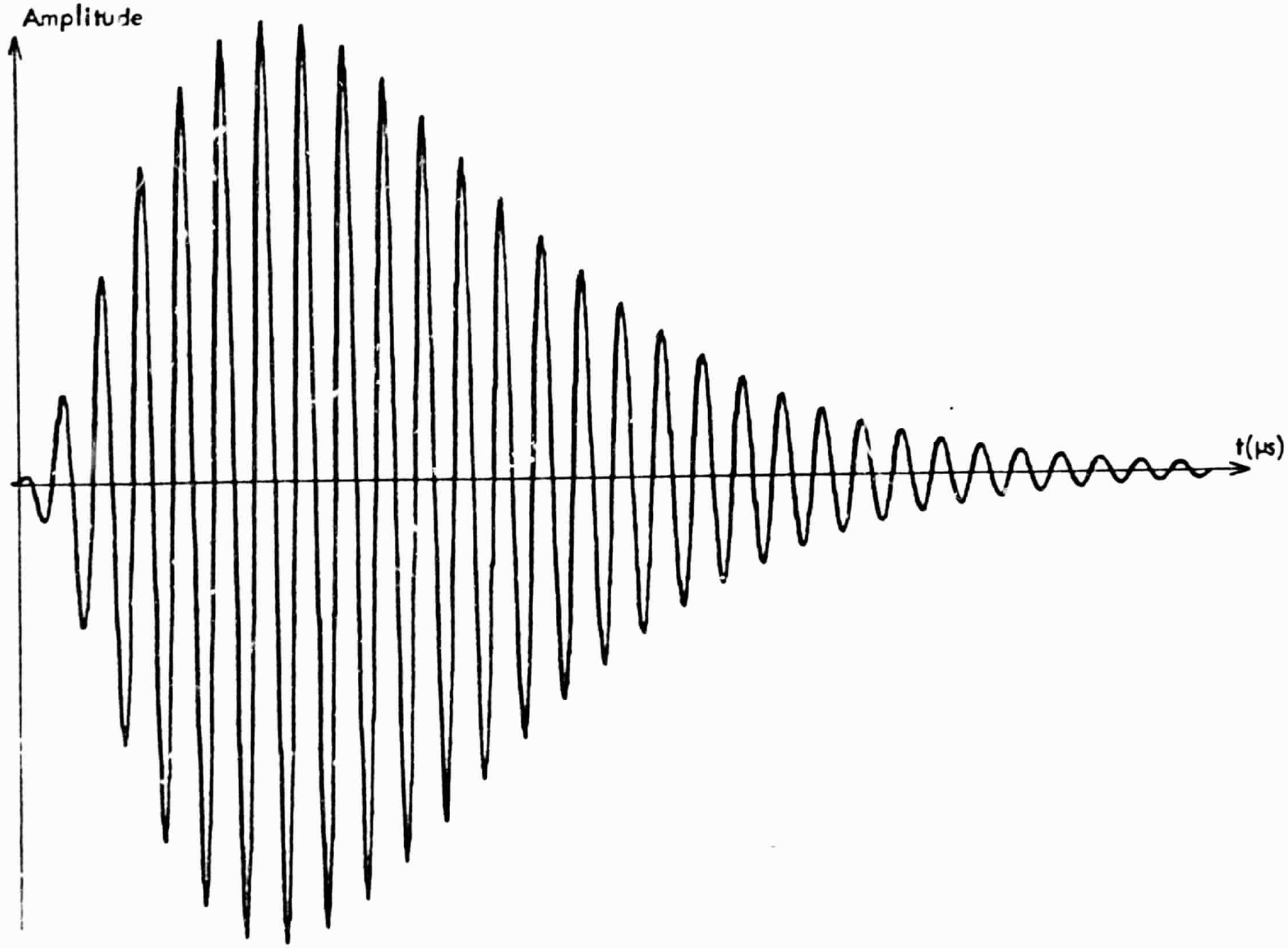

Figure 2-1. Loran-C Transmitted Waveform. Frequency of carrier is 100 KHz.

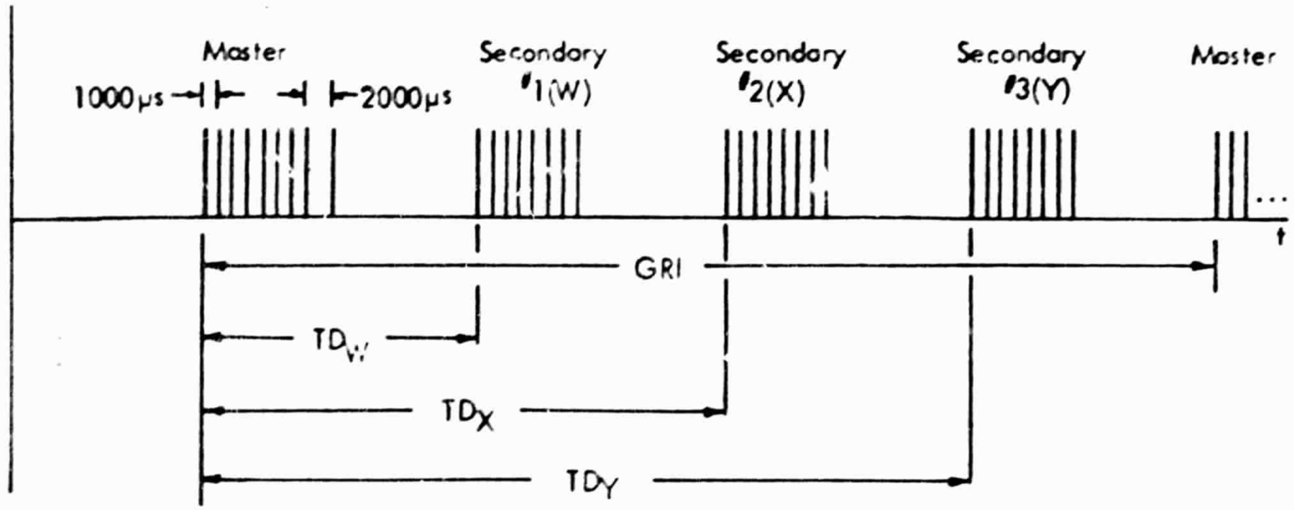

Figure 2-2. Transmission Format of Loran-C Stations in a Given Chain. The amount of time between the master and each of the secondaries, defines a time-difference (TD) for that master-secondary pair.

natural form of navigation information is the user's location is relative to a reference point. A hyperbolic system only provides an abstract set of coordinates defined by the particular setup of stations.

3. Lines-of-Position. By measuring the difference in time of arrival of a set of pulses from two stations, a TD number is found. Referring to Figure 2-3, a TD number is a point on the locus of points all having the same value. Denoting the distances from each of stations to the point as d_1 and d_2, and the constant difference as 2a, then

$$d_1 = \sqrt{(x+y)^2 + y^2} \qquad (2\text{-}1)$$

and

$$d_2 = \sqrt{(x-c)^2 + y^2} \qquad (2\text{-}2)$$

Since the difference of the distance is constant, i.e., $d_1 - d_2 = 2a$, then

$$\sqrt{(x+c)^2 + y^2} - \sqrt{(x-y)^2 + y^2} = 2a \qquad (2\text{-}3)$$

Rearranging and squaring both sides

$$(x+c)^2 + y^2 = 4a^2 + 2a\sqrt{(x-c)^2 + y^2} + (x-c)^2 + y^2 \qquad (2\text{-}4)$$

By further rearranging equation (2-4)

$$(a^2 - c^2)x^2 - a^2y^2 = a^2(a^2 - c^2) \qquad (2\text{-}5)$$

substituting $c^2 = a^2 + b^2$, the standard hyperbolic equation is produced. That is

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \qquad (2\text{-}6)$$

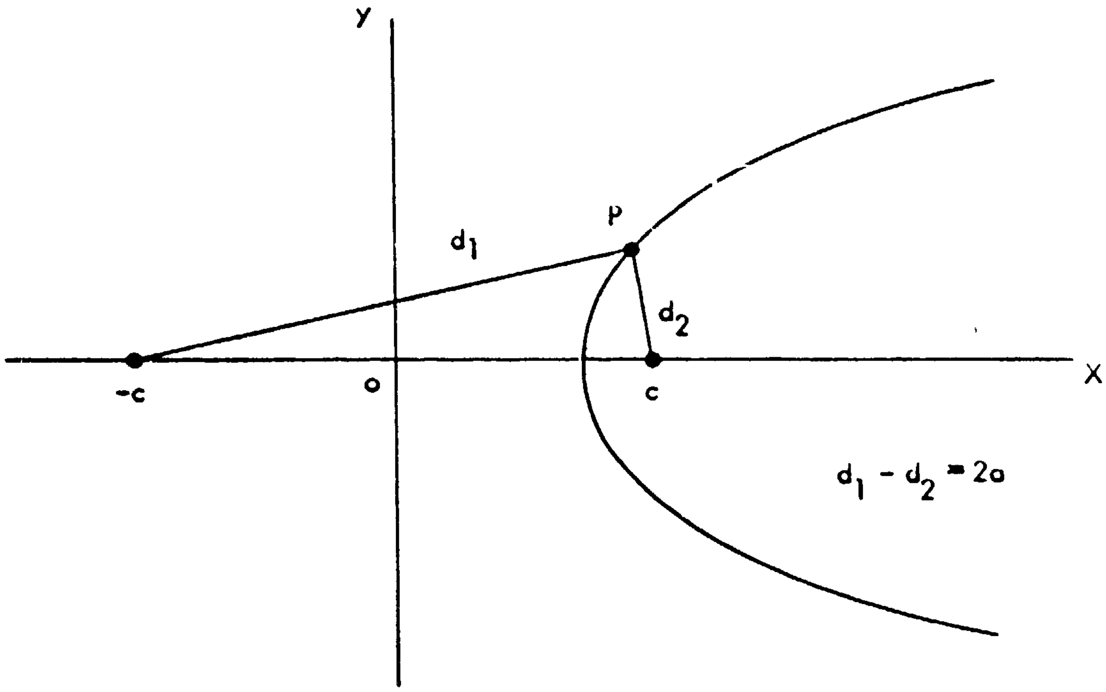

Figure 2-3. The TD Value Received at Point P From the Loran Stations at (-c,o) and (c,o) is on the Locus of Points with Constant TD of 2a.

Point a represents the x-axis crossing of the hyperbola and point b is the conjugate axis length such that the asymptote of the hyperbola is $y = \frac{b}{a}x$.

Each hyperbolic locus of points is a line-of-position (LOP). Since one LOP does not uniquely define the receiver's position, a second master-secondary pair must be used to define a second LOP; the intersection of two or more LOPs defines the receiver's position, as shown in Figure 2-4.

C. Method of computing time-differences. Reference for the following discussion will be made to Figure 2-5. The signal transmitted by the master station (M) is radiated in all directions. The time it takes for the signal to reach a given point is related to the speed of light (approximately 300 meters per μs) and to a lesser extent, the medium through which the signal travels. The secondary stations always transmit after the master; the time interval between the transmission of the master signal and the secondary signal is equal to the amount of time it would take the master's signal to reach the secondary (baseline time) plus a coding delay. Coding delays are introduced at the secondaries to ensure that no two stations will overlap anywhere in the coverage region. At the receiver position, there will be two signals present, the master and the secondary separated by a certain amount of time. The amount of time can be expressed mathematically as:

$$TD = \beta + \Delta + T_S - T_M \qquad (2\text{-}7)$$

where TD is the time difference in arrival of the master and secondary signals, β is the one-way baseline time between the master and secondary, Δ is the secondary coding delay inserted by the secondary, T_S is the one-way baseline time between the secondary station and the receiver, and T_M is the one-way baseline time between the receiver and the master station. The baseline time T_M is subtracted from the above equation to account for the receiver's displacement from the master station. Equation (2-7) contains the difference of x and y terms in the form of T_s and T_M plus additional constants and is thus in the form of a hyperbolic equation.

The baseline quantity β and the coding delay are generally known from the values set up during the installation of the Loran-C chain. The baseline quantity β can be accurately measured and the coding delay is chosen to provide reasonable distribution of the secondary signals throughout the group repetition interval. These two quantities are usually combined and published in the data for a particular chain by the U. S. Coast Guard [16]. The two baseline times from the receiver, T_S and T_M, are unknown and must be calculated before they can be applied to the TD equation. This involves computing

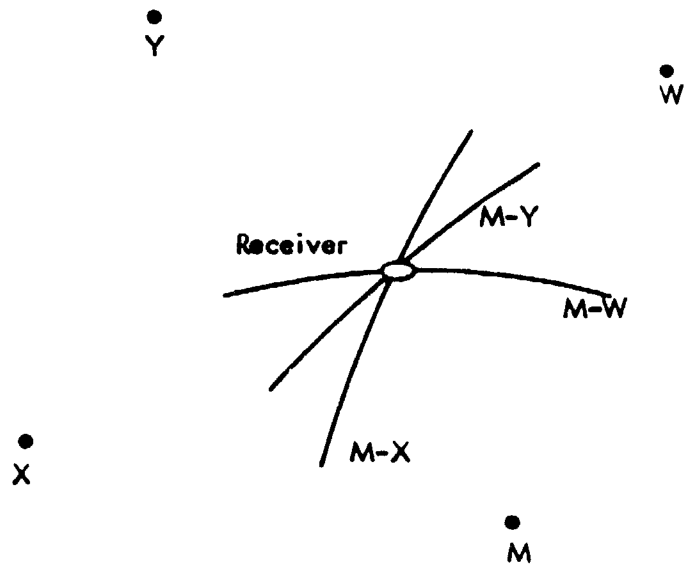

Figure 2-4. TDs Received from the Various Master-Secondary Pairs Define LOPs which All Intersect at the Receiver's Position. Each LOP corresponds to a particular TD number (in microseconds). The LOPs are actually hyperbolas with their corresponding master-secondary pair of stations as foci.

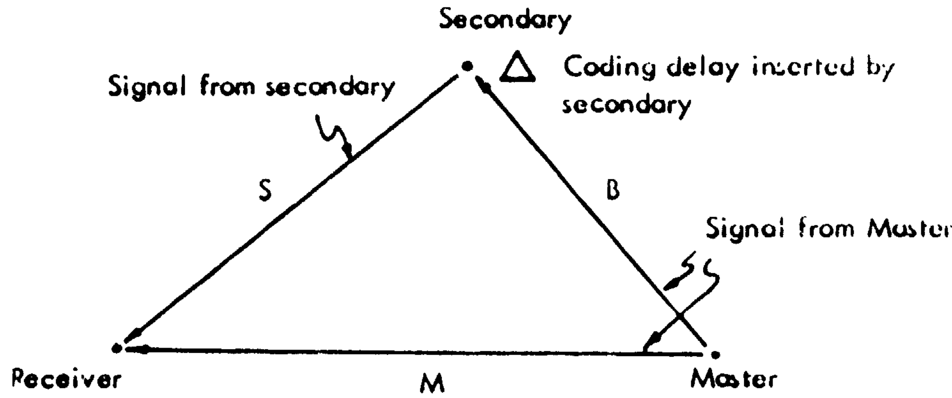

Figure 2-5. The TD Reading at the Receiver is Formed by the Amount of Time the Signal Travels Between the Master and Secondary, Plus the Coding Delay, Plus the Amount of Time the Signal Travels from the Secondary to the Receiver, Minus the Amount of Time the Signal Traveled from the Master to the Receiver.

the length between the receiver and the Loran-C stations and computing how much time it would take for the Loran-C signal to traverse this distance. Being able to do this is the most important part of a TD prediction algorithm.

Computing the amount of time required for an electromagnetic wave to travel from one point to another involves two factors: calculating the exact baseline distance between the two points taking into account the non-spherical nature of the earth, and applying corrections to the velocity of the signal as it passes through the medium around the baseline. The conductivity of the earth and effect of the air have the greatest impact on the velocity of propagation.

A method of calculating baseline length between two points on the surface of the earth will now be presented. As shown in Figure 2-6, the earth is approximated by an ellipsoid with major (equatorial) radius, a, and minor (polar) radius, b. The usual coordinate frame for defining a point on the surface of the earth is the geodetic spherical coordinate system where the latitude is represented by Φ and longitude is represented by λ. The geodesic arc-length between two points on the earth may be computed by the following procedure which calculates the arc-length on a plane through the center of the ellipsoid [17]. One begins by defining Φ as the geodetic latitude of the receiver and Φ_i as the geodetic latitude of the second point (Loran-C transmitter) and $\Delta\lambda$ as the difference in longitude between the receiver and second point. The procedure for computing the arc length is to consider a sphere of radius a circumscribing the earth. By projecting the latitude of a point on the earth onto the sphere, the point on the sphere has a parametric latitude denoted by β [18]. The parametric or reduced latitude is computed as:

$$\tan\beta = (1 - f)\tan\Phi \tag{2-8}$$

$$\tan\beta_i = (1 - f)\tan\Phi_i \tag{2-9}$$

where the subscript i represents the second point and f is the flattening of the ellipsoid:

$$f = \frac{a - b}{a} \tag{2-10}$$

The generalized direction cosines of the projected point are:

$$C_1 = \cos\beta_i \sin\Delta\lambda \tag{2-11}$$

$$C_2 = \cos\beta\sin\beta_i - \sin\beta\cos\beta_i\cos(\Delta\lambda) \tag{2-12}$$

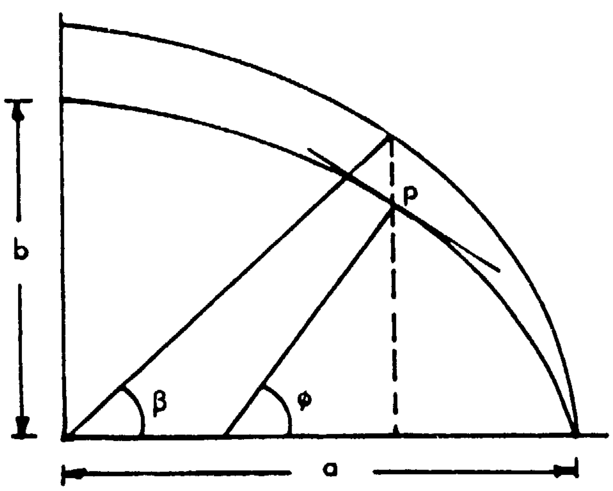

Figure 2-6. The Earth is Approximated by an Ellipsoid. A sphere is circumscribed around the ellipsoid with the major radii equal. Then the point p on the earth is projected onto the sphere.

$$C_3 = \sin\beta\sin\beta_1 + \cos\beta\cos\beta_1\cos(\Delta\lambda) \qquad (2\text{-}13)$$

The bearing angle, with respect to true north, from the receiver to the second point, measured at the receiver is:

$$\tan\Psi = \frac{C_1}{C_2} \qquad (2\text{-}14)$$

The approximate angle from the receiver to the second point, in a plane through the center of the ellipsoid is:

$$\tan\Theta = \frac{C_2\cos\Psi + C_1\sin\Psi}{C_3} \qquad (2\text{-}15)$$

The geodesic arc length between the receiver and the Loran-C transmitter may be found from

$$\rho = a\Theta \frac{\alpha f}{4} (mu + nv) \qquad (2\text{-}16)$$

where:

$$m = (\sin\beta + \sin\beta_1)^2$$

$$n = \left(\frac{\sin\beta - \sin\beta_1}{\sin\Theta} \right)^2 \qquad (2\text{-}17)$$

$$u = \left(\frac{1 - \cos\Theta}{\sin\Theta} \right) \left(\frac{\Theta - \sin\Theta}{\sin\Theta} \right)$$

$$v = (1 + \cos\Theta)\ (\Theta + \sin\Theta)$$

To calculate Loran-C time differences, it is necessary to do this procedure twice; once for the receiver to master station arc length and once for the receiver to secondary station arc length. The units of ρ depend on the units used to specify the radius of the earth, a. If the units of a are in microseconds (300 meters = 1 μs), then ρ gives directly the amount of time needed for the Loran-C signal to travel from one point to another. This method uses an oblate spheroid model of the earth to approximate the geodesic. Over distances typical of those involved with Loran-C work, this usually is adequate. In a generalized development of geodesics on an ellipsoid by Holmstrom, a three-dimensional model of the earth was developed allowing the three axes to be unequal. Using an oblate spheroid case where the equatorial axes are equal, the demonstrated error over a 482 km geodesic was 0.82 m with a 0.035 second of arc error [19].

Although the above procedure may be used to calculate Loran-C time differences, the assumption made is that the Loran-C signal will travel over the baseline path with a constant velocity, independent of the adjacent medium. In practical Loran-C work, the Loran-C signal travels over media which can have widely varying characteristics. Except for all-water paths, the signal travels over a surface of inhomogeneous conductivity and dielectric constants and also one which has irregular surfaces. There have been many attempts to produce a working model of the earth's surface in order to resolve the problem of predicting the signal phase delay, but such attempts are very difficult in light of the non-spherical and the irregular surface impedance nature of the earth. In an integral equation model of a nonhomogeneous, irregular earth proposed by Samaddar, predicted secondary phase delays of 2.5 to 3.5 μs over a path length of 300 to 400 km were calculated which compare favorably to measured values [20]. The difficulty in applying such models lies in getting an accurate profile of the section of the earth over which the signal will pass. The numerical calculations involved in solving the integral equations are complex and yield a phase factor correction of only a few microseconds More general impedance models are typically employed, such as with the Defense Mapping Agency[3]

The total time difference equation including secondary phase factor corrections discussed above is:

$$TD = \frac{\nu}{C}(P_S - P_M) + (\epsilon_S - \epsilon_M) + (\frac{\nu L}{C} - \Delta) \qquad (2\text{-}18)$$

where P_S and P_M are the calculated baseline lengths to the Loran-C secondary and master stations, respectively, using equation (2-16), ν is the index of refraction over the geodesic, C is the free-space speed of light, L is the baseline length between the master and secondary stations, Δ is the secondary coding delay, and ϵ_S and ϵ_M are the secondary phase factor corrections for the secondary and master geodesics, respectively.

---

[3]The Defense Mapping Agency computes corrections for overland phase retardation based on a data base of calibrated stations.

Appendix I lists a FORTRAN-IV program run on an IBM 370/158 to calculate Loran-C time differences. Various constants, including the geodetic latitudes and longitudes of the Loran-C master and secondary stations, the master-to-secondary baseline time delays, the secondary coding delays, and earth constants are all stored internal to the program. The program accepts the coordinates of the receiver as input. The oblate spherical earth model discussed above is used, however, no secondary phase factor corrections are applied. Figure 2-7 shows the results of using this program with several points along with actual time difference values calculated for these points using over-land phase retardation corrections [21]. This shows a typical error of two to three microseconds. The main source of error in this program is the lack of input for the secondary phase factor correlations. It should be noted that the Defense Mapping Agency maintains phase factor corrections for only a few control points; thus their calculations are not always exact for the user's position.

Position	Control TDS	Calculated TDS
41°18'49.5"N 72°41'33.5"W	26477.02-X 44062.39-Y	26477.55-X 44058.64-Y
41°39'55.0"N 73°49'21.0"W	27140.17-X 44369.04-Y	27137.8-X 44366.42-Y
39°19'22.81"N 82°5'5[illegible].42"W	28539.50-X 42593.39-Y 56775.49-Z	28538.44-X 42593.52-Y 56777.75-Z

Figure 2-7. Comparison of TDs Calculated by Defense Mapping Agency [21] (Control TDs) vs. TDs Calculated by Program in Appendix I for Three Different Locations. Loran-C Chain 9960 (U.S. Northeast) was used here.

III. CONVERTING TDs TO POSITION

The preceding chapter has dealt with how to compute the Loran-C time differences which may be expected at a given location. As will be seen, this is necessary for some iterative Loran TD-to-position solutions. Now, the inverse problem will be considered, that of converting received Loran-C time differences to more generalized position coordinates, such as, latitude and longitude. In this chapter, several different techniques will be examined to evaluate their simplicity, accuracy, and suitability for being used in a small microcomputer-based navigation system. Some methods of converting these first-order derived position coordinates into a range and bearing form simulating a VORTAC reference which is more familiar to pilots will also be discussed.

A. Conversion Methods Available. The time difference readings obtained from the Loran-C receiver are generally of little use to a pilot. Even if a pilot could become familiar with the time differences in certain areas through experience, using similar time differences for a different triad of stations or with a different chain would be meaningless. As shown in Figure 3-1, there is no simple relationship between the Loran-C lines-of-position and a geocentric grid coordinate system, such as latitude-longitude. Further, the spacing between successive LOPs varies with the position in the coverage region; at large distances between stations, a change of 10 μs might relate to a linear change of 10 km, while at a distance close to the station, a change of 10 μs might correspond to a linear change of 1 Km. Thus, navigating by Loran-C time differences alone is not a very suitable method.

The classic method of relating time differences to a more universal set of coordinates is through the use of charts or tables. While this may suffice for maritime services (for which such charts and maps are made) where craft velocity is low, it is very unsuitable for aircraft use. Ideally, the navigation unit should allow the pilot to instruct the unit what course the pilot wants to fly, and the unit will then provide the pilot with course guidance information as the flight progresses without the pilot having to derive course guidance from maps or tables. For this reason, it is desired to calculate automatically the actual position in a universal set of coordinates from the Loran-C time differences. This information may then be used for input to a course director computer. The course direction computer would then drive a display indicating whether or not the aircraft is on course, and the pilot would then simply maintain an on-course indication.

There are many means available for converting Loran-C time differences to latitude and longitude. These may be divided into two basic processes, those which require repetitive calculations to arrive at the final result, and those which use a direct or closed-form solution to do the conversion.

ORIGINAL PAGE IS
OF POOR QUALITY

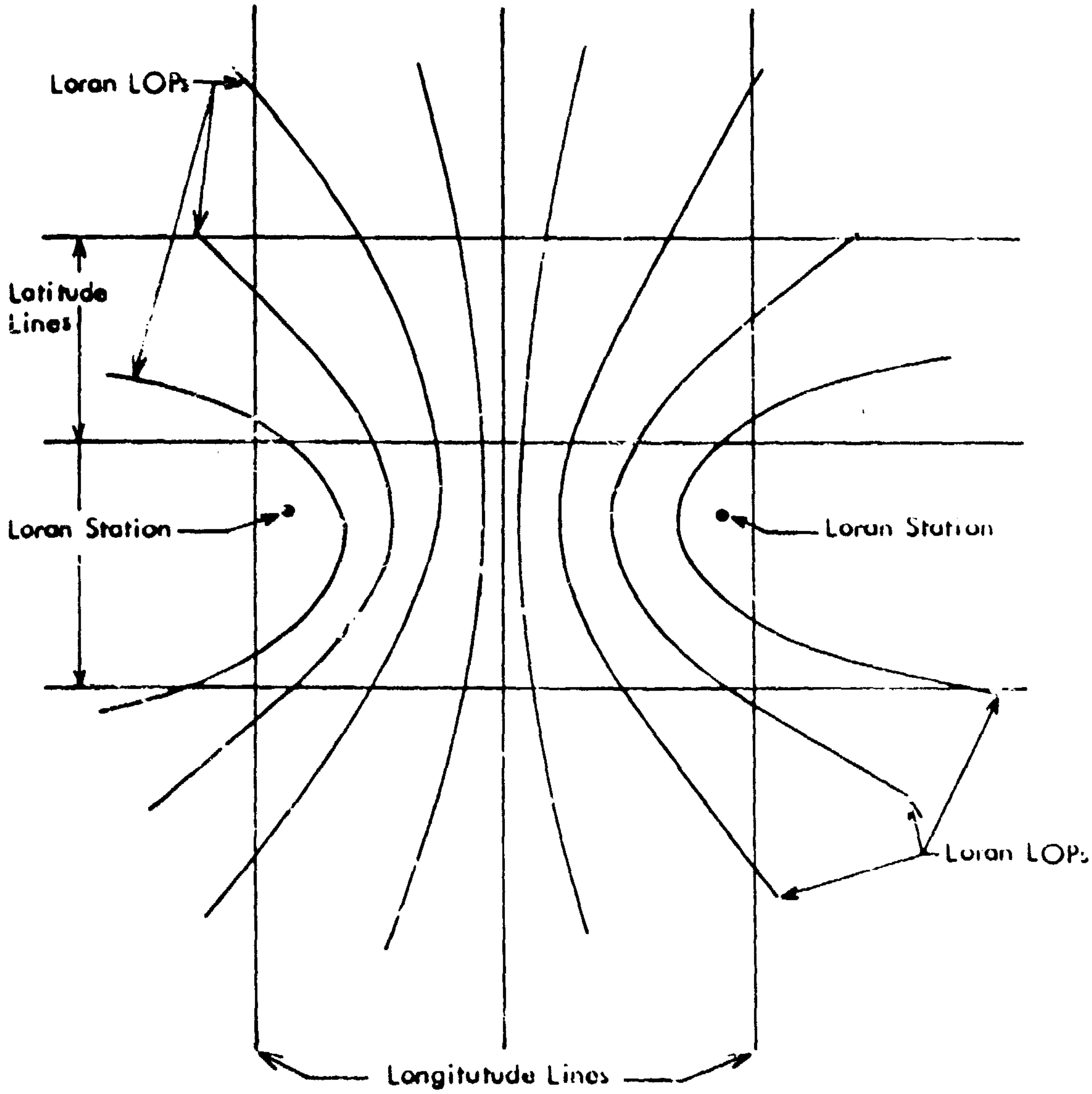

Figure 3-1. Relationship Between Loran-C Hyperbolic LOPs and Geocentric Grid.

An iterative TD-to-position algorithm functions by assuming that the receiver is at a given location, then comparing the position determined by the received TDs to the assumed position. The TDs for the assumed position are calculated before the comparison and then the assumed position is adjusted so as to minimize the error in comparison of the two sets of TDs. A way of mechanizing this is shown in Figure 3-2, where for a given triad of Loran-C stations, a table of TD values is generated spanning the coverage region of the triad, and another table is generated to the corresponding positions for the TD table. When a set of TDs are received, a process of linear interpolation is done between the two tables to find the actual position. This could be done several times to increase the accuracy of the fix. Other ways of relating received TD values to assumed TD values will be discussed in Chapter III.C.

A non-iterative, or closed-form solution, calculates the present position of the receiver by setting up a set of spherical equations in such a way that the received TDs can be used as parameters in these equations, then the equations are solved for the unknown values representing the receiver's position. This is generally complicated because spherical equations have angles as their arguments, and relationships between position on a geocentric grid and these spherical angles must be found. Once this is done, the equations are solved, generally with a computer, and the resulting functions are related back to the geocentric grid to find the latitude and longitude of the receiver corresponding to the received TDs.

B. Analysis of Conversion Methods. There are many specific TD-to-position methods to choose from, and there may be several that perform in a similar fashion. At this point, several criteria will be developed to select one for use in the microprocessor controlled navigator unit.

The overall accuracy of any particular method is important; some conversion methods are simple from a computational point of view, but suffer from poor accuracy and possible instabilities in their calculation. For aircraft use, the conversion method should have an accuracy of better than one kilometer; accuracies on the order of 0.1 nm should be more typical. This assumes that the received TDs correspond exactly to the actual position; then the TD-to-position conversion will converge to this point with a possible error of 0.1 nm. This confidence interval assures that a pilot will be able to reach a given navigation fix or airport with good accuracy.

A second limiting factor on choosing a conversion method is how much computational time will be required. For maritime use, where craft velocities are low, computation times of one minute, or more, are reasonable. However, for aviation purposes, a considerably faster position computation is needed. Most general purpose microprocessors on the market today, do not lend themselves very well to arithematic calculations; typically, there are many involved calculations that

42500, 56700		39 9 53, 82 11 31
42510, 56710		39 10 53, 82 10 40
42520, 56720		39 11 54, 82 9 50
42530, 56730		39 12 55, 82 9 1
42540, 56740		39 13 55, 82 8 10
42550, 56750	42554.3, 56758.4	39 14 56, 82 7 20
42560, 56760		39 15 56, 82 6 30
42570, 56770		39 16 56, 82 5 41
42580, 56780		39 17 56, 82 4 51
42590, 56790		39 18 56, 82 4 2

Figure 3-2. A Method of Converting TDs to Position By Comparing the Received TDs to the Nearest Corresponding TD Pair in A TD-position Table, Then Using Linear Interpolation to Find the Position.

need to be made in a TD-to-position conversion. Thus, a conversion alogrithm should involve a minimum of complex type mathematical calculations, particularly transcendental functions, so that the microprocessor system can perform a TD-to-position conversion in less than one second.

Finally, the size of the conversion algorithm needs to be considered. In most microprocessor applications, it is desirable to keep the memory size low to avoid excessive power drain and other operational problems that can arise in large memory systems. Even on large computer systems, a small conversion program is easier to store, and a program that requires a small amount of memory is easier to manage in main storage and will often run faster for that reason alone.

C. An Algorithm for Microcomputer Use. Several possible TD-to-position conversion methods will now be analyzed using the concepts developed above. Iterative conversions are examined first.

As mentioned briefly in Chapter III-1, after receiving a pair of Loran-C time differences, it is necessary to compute the amount and direction in which the initial position should be adjusted. A property of hyperbolic navigation systems is helpful in this problem (Figure 3-3); referring again to the TD equation $TD = \beta+\Delta+T_S-T_M$, it can be seen that as the receiver is moved closer to the master (farther from the secondary), T_S will increase and T_M will decrease, and the TD value for that master-secondary pair will increase. Thus if the difference between the assumed position and the measured TDs is positive, the new estimate of position should be made closer to the master station to increase the value of the calculated TD and decrease the result of the subtraction of the two TDs. This process is illustrated in Figure 3-4.

Note that this algorithm could take many iterations to reach an acceptable estimate of actual position. The amount of position change to be made with respect to the magnitude of the TD error must be carefully chosen for the chain and particular stations used. Depending on the actual position relative to the Loran-C stations and any baseline extensions between stations, this routine could suffer from a large radius of convergence and poor convergence time.

Another method of calculating the actual position based on the received TDs is to assume an initial reference point and calculate the TDs at that point as before. Then the measured TDs are compared with the TDs based on the assumed position. The assumed position is then moved to a new position based on the magnitude and sign of the TD error. This procedure differs from the previously discussed case in that an attempt is made to relate TD errors to position errors; i.e., the angle of the LOP passing through the receiver position is broken down to cartesian components and the TD error is used as a weighting factor along with the LOP components to calculate the new position.

ORIGINAL PAGE IS
OF POOR QUALITY

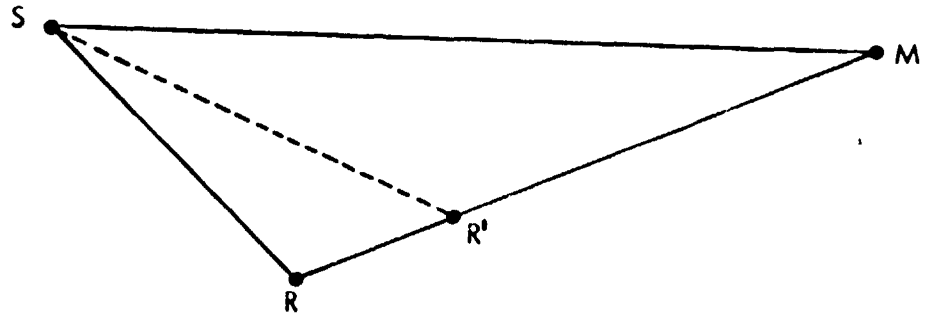

Figure 3-3. By Moving the Receiver Position (R) Closer to the Master (R') Along the Master–Receiver Baseline, the Assumed TD May Be Made to More Nearly Correspond to the Received TD Provided the Initial TD was too high.

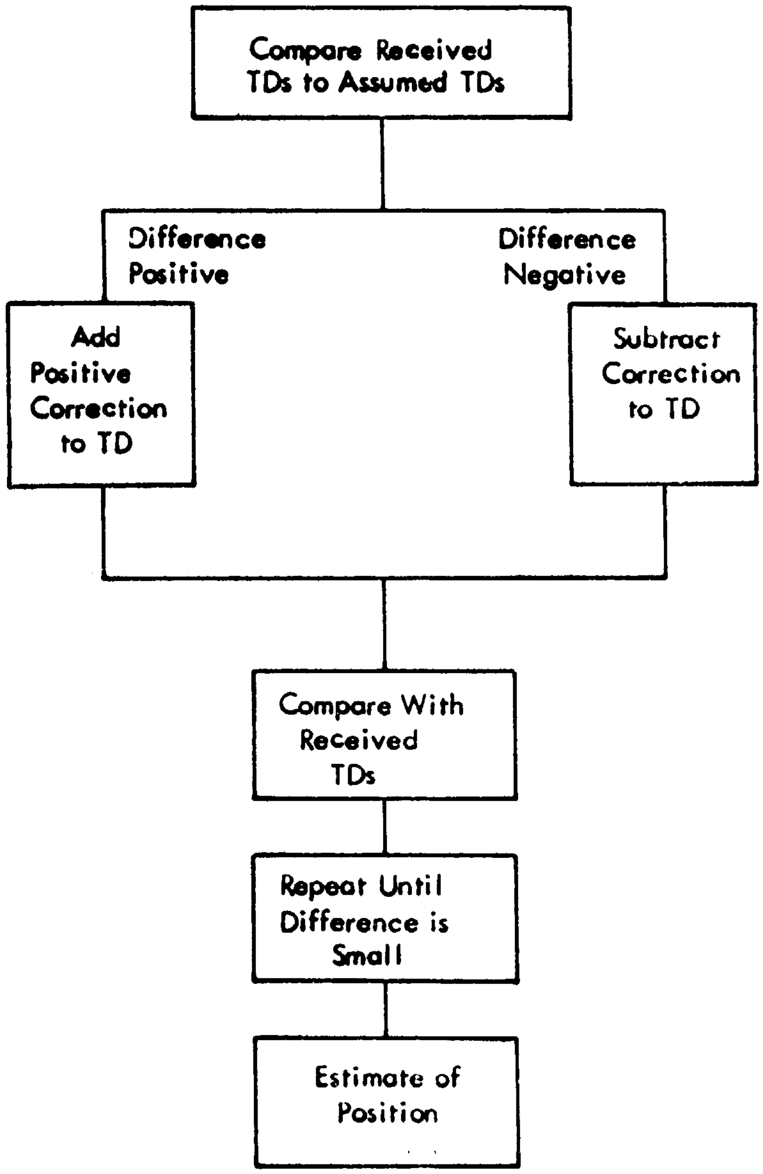

Figure 3-4. A Method of Converting TDs to Position by Minimizing the Comparison Error by Moving the Assumed Position Closer to or Farther from the Master.

This process is iterated until the TDs based on the adjusted position agree with the measured TDs within an acceptable range.

The mathematics of this algorithm will be more fully developed at this point. Referring to Figure 3-5, the tangent to the hyperbolic LOP passing through the receiver position is:

$$\Psi_{SM} = \frac{\Psi_S + \Psi_M}{2} \qquad (3\text{-}1)$$

where Ψ_S and Ψ_M are the bearings, from true north, of the geodesic arcs from the receiver to the secondary and master stations, respectively. The tangent to the hyperbola also bisects the angle between the two arcs to the Loran-C stations, as shown in Figure 3-5. A relationship between TD error and position error is found by considering a vector whose magnitude is:

$$|\nabla| = \frac{2}{c}\sin\left(\frac{\Psi_S - \Psi_M}{2}\right) \qquad (3\text{-}2)$$

The north and east components of $|\nabla|$ are [22]:

$$\alpha = \frac{1}{c}(\cos\Psi_S - \cos\Psi_M) = -\nabla\sin\Psi_{SM}$$

$$\gamma = \frac{1}{c}(\sin\Psi_S - \sin\Psi_M) = \nabla\cos\Psi_{SM} \qquad (3\text{-}3)$$

The TD errors are calculated as follows:

$$\delta T_A = T_{TA} - T_{OA}$$

$$\delta T_B = T_{TB} - T_{OB} \qquad (3\text{-}4)$$

where T_{TA} and T_{TB} are the TDs corresponding to the assumed position, and T_{OA} and T_{OB} are the measured TDs (a position fix requires the use of at least two secondary stations, resulting in two time difference measurements). The new estimate of position is:

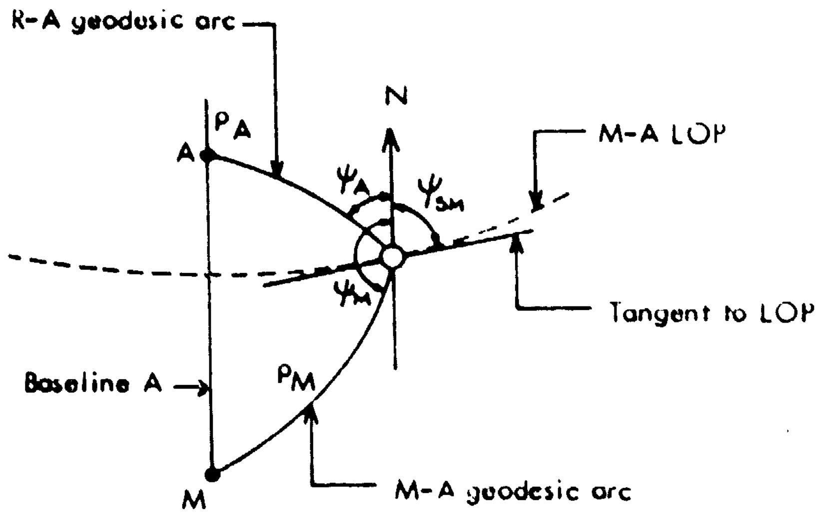

Figure 3-5. Geometry of Master (M), Secondary (A) and Receiver (R) Position. The tangent to the LOP bisects the angle formed between the geodesic arc from the receiver to the master and secondary.

ORIGINAL PAGE IS
OF POOR QUALITY

$$\phi_2 = \phi_1 + \frac{1}{aD}(\delta T_A \gamma_2 - \delta T_B \gamma_1)$$

$$\Delta\lambda_2 = \Delta\lambda_1 + \frac{1}{aD\cos\phi}(\delta T_B \alpha_1 - \delta T_A - \delta T_A \alpha_2) \qquad (3\text{-}5)$$

where

$$D = \alpha_1\gamma_2 - \alpha_2\gamma_1 = V_1V_2\mathrm{Sin}(\Psi_{SMA} - \Psi_{SMB}) \qquad (3\text{-}6)$$

and the subscripts 1 and 2 refer to quantities associated with secondary stations A and B, respectively.

Analysis of this algorithm showed that it involves fairly extensive calculations; as many iterations are required, the TD prediction algorithm is called many times and can cause a severe time and memory penalty.

For use in a navigation unit, once a position fix has been determined, succeeding fixes will be close to the preceding ones, a fact which can simplify position fix calculations. Many iterative algorithms cannot make use of this fact, since their first correction may overshoot the actual position and may be along a direction which is not coincident with the navigation track.

A computer program developed by the Naval Oceanographic Office [23] uses a technique whereby a set of Loran-C TDs may be converted to latitude and longitude if a starting point close to the actual position is known. This process uses the baseline lengths to the stations (expressed in microseconds) to set up a homogeneous set of transcendental equations which are then solved for unknown corrections applied to the initial position. If the initial position is more than 10 to 20 kilometers from the actual position, cyclic ambiguities arise in the solution to the set of equations, thus a series of iterations on the initial position may be necessary before the actual position may be determined. This program will converge directly on the correct position, thus eliminating the problem of having a large radius of converge, common with the previously discussed cases. Although this program has proven to be very accurate for use on an IBM 370 [24], a considerable number of calculations are necessary to obtain the solution. For use in a small computer, this would result in a large amount of memory for the calculations. The solution is faster than the previous case; however, since only one iteration is necessary once an initial point close to the actual receiver position has been determined.

Several other iterative-type solutions are possible and have been investigated. One set of conversion methods in particular has been developed which solves the TD-to-position on the transverse Mercator xy grid and are fairly simple [25]. However, to use this, a large set of conversion equations must be solved to convert from the UTM (Universal Transverse Mercator) coordinates to the standard lat/long coordinates, which involves the solution of many transcendental equations.

A non-iterative, explicit solution to the TD-to-position problem has often been envisioned as an ideal substitute to the above algorithms. Explicit solutions are complicated by having to deal with the non-spherical nature of the earth and also with the non-constant propagation properties of the Loran-C signals, as discussed in Chapter II. These effects can be introduced as local correction quantities in the iterative solution once the iterated position becomes close to the actual position. However, with explicit solutions, these corrections must take the form of a general model which represents a large area. Thus, an explicit solution can be simple if precise accuracy is not necessary.

Non-iterative solutions may take several different forms, however, most form a relation between the received time differences and the corresponding distances from the receiver to the Loran-C stations. From here, these distances and angles between baseline arcs are used to form a final expression from which the receiver position is found [26].

An excellent method for converting TDs to position coordinates in the form of latitude and longitude was presented by Razin [27] and is presented here. Referring to Figure 3-6, the spherical arc angles to the secondary stations are:

$$\Theta_{xs} = \Theta_{ms} + P_x$$

$$\Theta_{ys} = \Theta_{ms} + P_y \qquad (3\text{-}7)$$

where:

$$P_x = \frac{\nu}{R}(T_x - T_{cx}) - \Theta_{mx}$$

$$P_y = \frac{\nu}{R}(T_y - T_{cy}) - \Theta_{my} \qquad (3\text{-}8)$$

and $\nu = \nu_0/n$, the velocity of light divided by the index of refraction over the surface, R is the radius of the sphere in Figure 3-6, T_x and T_y are the measured time differences received for stations X and Y, respectively, and T_{cx} and T_{cy} are the secondary coding delays for stations X and Y, respectively. Notice that in this case, the measured time differences are converted into spherical arc angles for the

ORIGINAL PAGE IS
OF POOR QUALITY

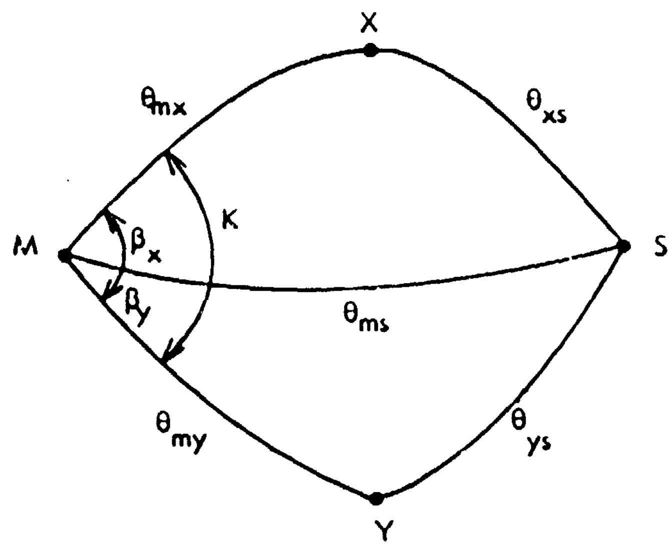

Figure 3-6. Geometry On a Sphere of a Loran-C Complex Consisting of Master (M), Secondary-X (X), Secondary-Y (Y) and the Receiver (S). The θ's are arc angles measured at the center of the sphere and the other angles are spherical angles.

receiver baselines to the Loran-C stations. Now taking the cosine of equation (3-7):

$$\cos\theta_{xs} = \cos\theta_{mx}\cos P_x - \sin\theta_{ms}\sin P_x$$

$$\cos\theta_{ys} = \cos\theta_{ms}\cos P_y - \sin\theta_{ms}\sin P_y \tag{3-9}$$

From the properties of arc lengths on a sphere, these equations may also be expressed as:

$$\cos\theta_{xs} = \cos\theta_{mx}\cos\theta_{ms} + \sin\theta_{mx}\sin\theta_{ms}\cos\beta_x$$

$$\cos\theta_{ys} = \cos\theta_{my}\cos\theta_{ms} + \sin\theta_{my}\sin\theta_{ms}\cos\beta_y \tag{3-10}$$

Equating equations (3-9) and (3-10) results in

$$\tan\theta_{ms} = \frac{\cos P_x - \cos\theta_{mx}}{\sin P_x + \sin\theta_{mx}\cos\beta_x} \tag{3-11}$$

From Figure 3-6,

$$\cos\beta_y = \cos(K-\beta_x) = \cos K\cos\beta_x + \sin K\sin\beta_x \tag{3-12}$$

and β_x may be solved in terms of the known quantities:

$$\cos\beta_x = \frac{u_3u_1 + u_2\sqrt{u_1^2 + u_2^2 - u_3^2}}{u_1^2 + u_2^2} \tag{3-13}$$

where:

$$u_1 = a_x\cos K - a_y$$

$$u_2 = a_x \sin K$$

$$u_3 = a_y \frac{\sin P_x}{\sin O_{mx}} - a_x \frac{\sin P_y}{\sin O_{my}}$$

(3-14)

$$a_x = \frac{\cos P_x - \cos O_{mx}}{\sin O_{mx}}$$

$$a_y = \frac{\cos P_y - \cos O_{my}}{\sin O_{my}}$$

$$b_x = \frac{\sin P_x}{\sin O_{mx}}$$

The angle O_{ms} is found then by:

$$O_{ms} = \tan^{-1} \frac{a_x}{b_x + \cos\beta_x}$$

(3-15)

and equation (3-7) may be used to find O_{xs} and O_{ys}.

To find the latitude and longitude of the present position, the spherical arc angles just calculated must be converted to spherical latitude and longitude. This process is rather complicated, but the results are a series of constants multiplied by the arc angles. These constants need be computed once for each set of stations used and may then be stored in the navigation computer. Let ϕ_m, ϕ_x and ϕ_y be the latitudes of the master, secondary-x and secondary-y and O_m, O_x and O_y be the longitudes of the master, secondary-x and secondary-y. The receiver's position is denoted by ϕ and O. The following relations on a sphere were developed by Fell [28]:

$$\cos\phi_x \sin O_x (\cos\phi \sin O) + \cos\phi_x \cos O_x (\cos\phi \cos O) + \sin\phi_x \sin\phi = \cos O_{xs}$$

(3-16)

$$\cos\phi_y \sin O_y (\cos\phi \sin O) + \cos\phi_y \cos O_y (\cos\phi \cos O) + \sin\phi_y \sin\phi = \cos O_{ys}$$

(3-17)

$$\cos\phi_m \sin\theta_m(\cos\phi\sin\theta) + \cos\phi_m\cos\theta_m(\cos\phi\cos\theta) + \sin\theta_m\sin\phi = \cos\theta_{ms} \tag{3-18}$$

These may be treated as three linear equations in three unknowns by letting $g=\cos\phi\sin\theta$, $f=\cos\phi\cos\theta$ and $h=\sin\phi$. The solutions of these equations are found by using θ_{xs}, θ_{ys} and θ_{ms} from equations (3-15), (3-16) and (3-17). These are:

$$f = C_1\cos\theta_{xs} + C_2\cos\theta_{ms} + C_3\cos\theta_{ys} \tag{3-19}$$

$$g = C_4\cos\theta_{xs} + C_5\cos\theta_{ms} + C_6\cos\theta_{ys} \tag{3-20}$$

$$h = C_7\cos\theta_{xs} + C_8\cos\theta_{ms} + C_9\cos\theta_{ys} \tag{3-21}$$

The constants are:

ORIGINAL PAGE IS OF POOR QUALITY

$$C_1 = (\cos\phi_m\sin\theta_m\sin\phi_y - \sin\phi_m\cos\phi_y\sin\theta_y)/D$$

$$C_2 = (\sin\phi_x\cos\phi_y\sin\theta_y - \cos\phi_x\sin\theta_x\sin\phi_y)/D$$

$$C_3 = (\cos\phi_x\sin\theta_x\sin\phi_x - \sin\phi_x\cos\phi_m\sin\theta_m)/D$$

$$C_4 = (\sin\phi_m\cos\phi_y\cos\theta_y - \cos\phi_m\cos\theta_m\sin\phi_y)/D$$

$$C_5 = (\cos\phi_x\cos\theta_x\sin\phi_y - \sin\phi_x\cos\theta_y\cos\theta_y)/D$$

$$C_6 = (\sin\phi_x\cos\phi_m\cos\theta_m - \cos\phi_x\cos\theta_x\sin\phi_m)/D$$

$$C_7 = (\cos\phi_m\cos\phi_y)(\cos\theta_m\sin\theta_y - \sin\theta_m\cos\theta_y)/D$$

$$C_8 = (\cos\phi_x\cos\phi_y)(\sin\theta_x\cos\theta_y - \cos\theta_x\sin\theta_y)/D$$

$$C_9 = (\cos\phi_x\cos\phi_m)(\cos\theta_x\sin\theta_m - \sin\theta_x\cos\theta_m)/D \tag{3-22}$$

$$\begin{aligned} D = {} & \sin\phi_x\cos\phi_m\cos\phi_y(\cos\theta_m\sin\theta_y - \sin\theta_m\cos\theta_y) \\ & - \sin\phi_m\cos\phi_x\cos\phi_y(\cos\theta_x\sin\theta_y - \sin\theta_x\cos\theta_y) \\ & + \sin\phi_y\cos\phi_x\cos\phi_m(\cos\theta_x\sin\theta_m - \sin\theta_x\cos\theta_m) \end{aligned} \tag{3-23}$$

Then the latitude and longitude of present position is:

$$\phi = \sin^{-1} h \tag{3-24}$$

$$O = \tan^{-1} \frac{g}{f} \tag{3-25}$$

The conversion of the received TDs to position using this method is illustrated in Figure 3-7. This algorithm uses a spherical model of the earth to find the lat/long of present position. By extending the treatment of converting the spherical arc angles to the corresponding geocentric coordinates, an elliptical model of the earth may be incorporated.

Using the Clarke 1866 spheroid, the major radius of the earth is R and the minor radius is r. Let P_o with latitude ϕ_o and longitude O_o, be approximately in the center of the coverage area of the master and two secondary stations. Then the center of the sphere osculating at P_o (see Figure 3-8) is:

$$X = (R - \sqrt{R^2 \sin^2\tilde{\phi}_o + r^2 \cos^2\tilde{\phi}_o})\cos\tilde{\phi}_o \sin O_o$$

$$Y = (R - \sqrt{R^2 \sin^2\tilde{\phi}_o + r^2 \cos^2\tilde{\phi}_o})\cos O_o \cos\tilde{\phi}_o$$

$$Z = (r - \frac{R}{r}\sqrt{R^2 \sin^2\tilde{\phi}_o + r^2 \cos^2\tilde{\phi}_o})\sin\phi_o \tag{3-26}$$

$$\tilde{\phi}_o = \tan^{-1}(\frac{R}{r}\tan\phi_o) \tag{3-27}$$

(Geodetic, or map latitude, is denoted by ϕ_o while geocentric latitude by $\tilde{\phi}$.) Also let p be a point on the spheroid with latitude ϕ and longitude O.

Its Cartesian coordinates are:

$$x = R\cos\tilde{\phi}\sin O$$

$$y = R\cos\tilde{\phi}\cos O \tag{3-28}$$

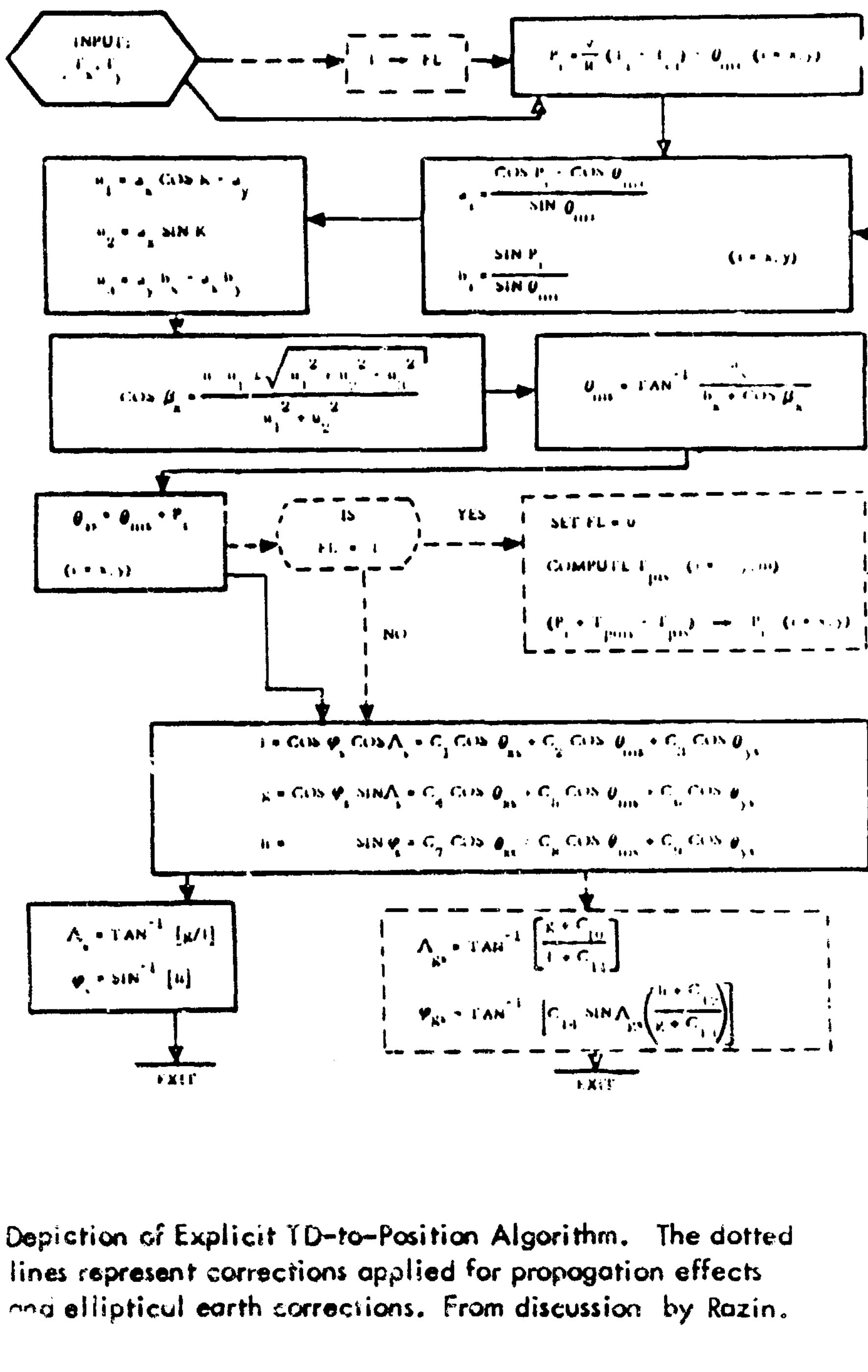

Figure 3-7. Depiction of Explicit TD-to-Position Algorithm. The dotted lines represent corrections applied for propogation effects and elliptical earth corrections. From discussion by Razin.

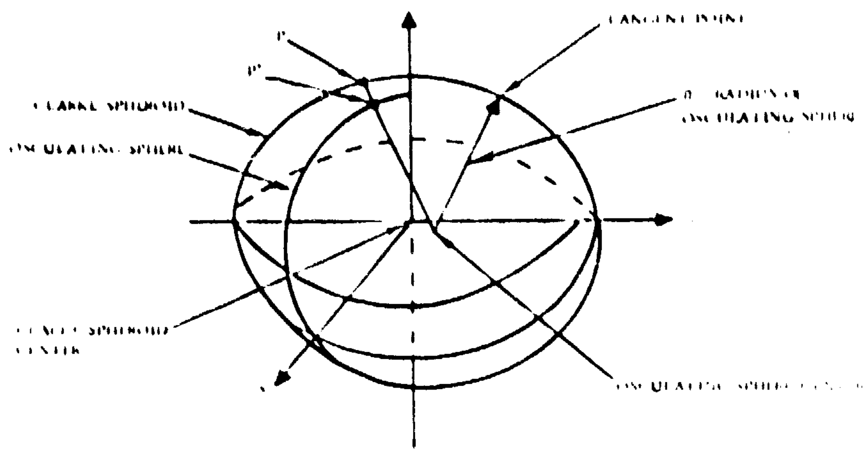

Figure 3-8. Mapping of Point P On the Clarke Ellipsoid to An Osculating Sphere Whose Surface is Tangent to the Point P.

$$z = r\sin\tilde{\phi}$$

$$\tilde{\phi} = \tan^{-1}\left(\frac{r}{R} \tan\bar{\phi} \right) \qquad (3\text{-}29)$$

Let p' be the image of p under the map from the spheroid to the sphere. Then the coordinates of p' are:

$$\frac{r_c(x-X)}{L} , \quad \frac{r_c(y-Y)}{L} , \quad \frac{r_c(z-Z)}{L} \qquad (3\text{-}30)$$

$$L = ((x-X)^2 + (y-Y)^2 + (z-Z)^2)^{1/2} \qquad (3\text{-}31)$$

$$r_c = \frac{R^2\sin^2\phi_0 + r^2\cos^2\phi_0}{r} \qquad (3\text{-}32)$$

The latitude and longitude of p' are:

$$\phi' = \sin^{-1}\left(\frac{z-Z}{L}\right)$$

$$\theta' = \tan^{-1}\left(\frac{x-X}{y-Y} \right) \qquad (3\text{-}33)$$

To implement this, the above procedure is done for each of the stations. In other words, letting ϕ_m, ϕ_x, ϕ_y, θ_m, θ_x and θ_y be the latitude and longitude of the master and two secondaries, then the image of these on the sphere are calculated giving ϕ_m', ϕ_x', ϕ_y', θ_m' θ_x' and θ_y' . These are then used to calculate the constants C_1 through C_9 as before.

With these nine constants and the calculated arc angles, the latitude and longitude must be mapped back onto the spheroid. The result is:

$$\theta_{gs} = \tan^{-1} \left(\frac{g + C_{10}}{f + C_{11}}\right)$$

$$\tilde{\phi}_{gs} = \tan^{-1} \left(C_{14}\sin\theta_{gs} \frac{h + C_{12}}{g + C_{13}}\right) \qquad (3\text{-}34)$$

ORIGINAL PAGE IS
OF POOR QUALITY

$$\phi_{gs} = \tan^{-1} \left(\frac{R}{r}\tan\tilde{\phi}_{gs}\right) \tag{3-35}$$

The other constants, C_{10} through C_{14} are:

$$C_{10} = \frac{X}{r_c}$$

$$C_{11} = \frac{Y}{r_c}$$

$$C_{12} = \frac{Z}{r_c}$$

$$C_{13} = C_{10}$$

$$C_{14} = \frac{R}{r} \tag{3-36}$$

Note that the computation of the C-constants is done once for each triad of Loran-C stations and in the navigation computer; only equations (3-7) through (3-20) are computed, and the C-constants are then used as multiplicative parameters. A FORTRAN-IV program implementing the non-iterative TD-to-position procedure is shown in Appendix II. A FORTRAN-IV program which calculates the C-constants using equations (3-21) through (3-35) is given in Appendix IV. It can be seen that the TD-to-position program is fairly short and does not involve very many complex calculations. Figure 3-9 shows a breakdown of the mathematical operations used.

D. Distance and Bearing Angle Computations. Having present position described in terms of latitude and longitude is of limited value for the pilot since the pilot will often need to refer to a chart or map to locate the present position relative to geographic features or other navigation aids. It is more helpful to have the present position given in terms of the range, or distance, and the azimuth, or bearing angle, to or from a waypoint, which could be the departure airport, the destination airport or some other reference point along the aircraft's route. The pilot may wish to use several different waypoints and these would in turn be stored in the navigation computer. Then, as the flight progresses, the computer would use the currently selected waypoint and the newest estimate of position to compute the range and azimuth. A way of computing this was presented in Chapter II for the development of computing Loran-C TDs given a known position. An alternate method is presented here which is simpler and better suited for these range and azimuth (also called rho/theta) calculations. Referring to Figure 3-10, the coordinates of

ORIGINAL PAGE IS
OF POOR QUALITY

Mathematical operations used	Number
Addition	15
Subtraction	9
Multiplication	25
Division	8
Sine	3
Cosine	5
Square root	1
Arc-tangent	3
Total	69

51 constants/variables used

Figure 3-9. Breakdown of Mathematical Operations and Variables Used in Explicit Coordinate Converter Program.

ORIGINAL PAGE IS
OF POOR QUALITY

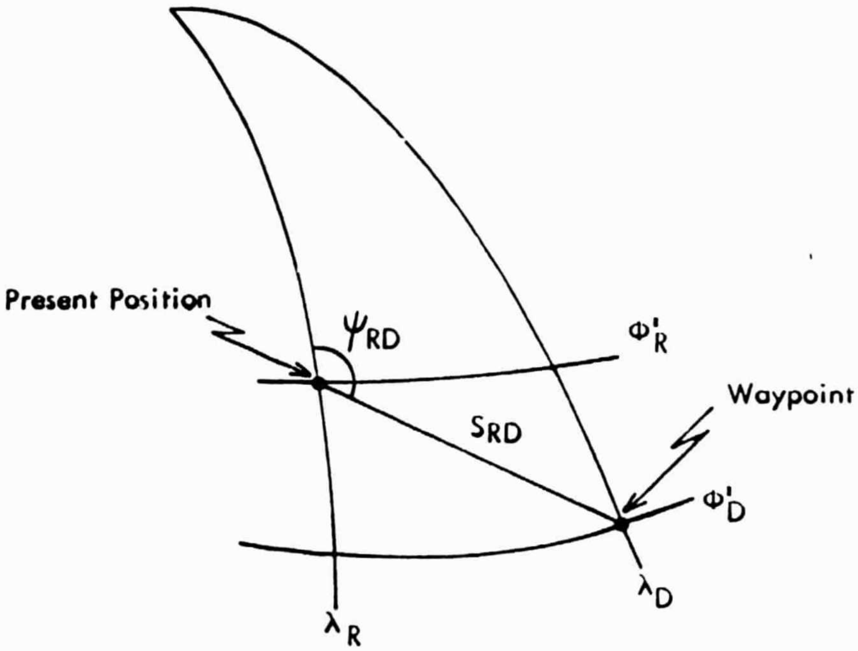

Figure 3-10. Geometry on Sphere Used for Range and Bearing Angle to Waypoint from Present Position Calculations.

ORIGINAL PAGE IS
OF POOR QUALITY

present position are denoted by ϕ_R and λ_R, and the coordinates of the waypoint are denoted by ϕ_w and λ_w for the latitude and longitude, respectively. The bearing angle to the waypoint is [29]:

$$\psi_{RD} = \tan^{-1}\left[\frac{(\lambda_R - \lambda_w)\cos(\frac{\phi_R + \phi_w}{2})}{\phi_w - \phi_R}\right] \tag{3-37}$$

and the range to the waypoint is:

$$S_{RD} = R\left[(\lambda_R - \lambda_w)\cos(\frac{\phi_R + \phi_w}{2})^2 + (\phi_w - \phi_R)^2\right]^{1/2} \tag{3-38}$$

where R is the radius of curvature of the earth. This may be computed for the mid-region of the coverage for the particular Loran-C stations used and stored as a constant. After choosing the midpoint of the coverage region, R is found by:

$$R = \frac{a^2\sin^2\phi_{mn} + b^2\cos^2\phi_{mn}}{b} \tag{3-39}$$

where a and b are the semi-major and semi-minor radii of the reference ellipsoid and ϕ_{mn} is the latitude of the mid-point.

E. Program for Coordinate Conversions. Table 3-1 lists trade-off considerations for the various TD-to-position methods. The non-iterative TD-to-position procedure outlined in Section C was found to be the most suitable of any for application to a microprocessor-based coordinate converter. As already mentioned, the most complex calculations, involving the C-constants, may be carried out on a central computer. Then the resulting C-constants are transferred to the microprocessor system. For any given triad of Loran-C stations, the same C-constants are used regardless of the aircraft's position.

The ellipsoidal model gives better results than the spherical model, as might be expected. The degree to which the ellipsoidal model does better depends on the choice of P_o in equation (3-25). This should be chosen to be in the center of the coverage region of the triad. This is most simply done by averaging the latitudes and longitudes of the master and two secondary stations. This average latitude and longitude is then used for P_o.

The direct TD-to-position algorithm described in Section C was tested for accuracy and to determine if it exhibits any singularities,

ORIGINAL PAGE IS
OF POOR QUALITY

TD-to-Position Method	Advantages	Disadvantages
Iterative method of adjusting position closer to or farther from master.	Simple to program	Could require large number of iterations to converge on result. Time consuming. Large radius of convergence.
Iterative method of adjusting position based on size of TD comparison error.	Requires fewer iterations than above. Small radius of convergence. Possible to insert local propagation corrections for better accuracy.	Time consuming - TD prediction routine called repeatedly. Requires more memory than above. Possible to have points of singularity in solution.
Direct one-iteration solution for assumed position close to actual position.	Provides high accuracy. Faster than above two. Result is given with predictable amount and direction of possible error from true position.	Much larger memory requirements than above two. Requires iterations or dead-reckoning if assumed position is far from actual position.
Direct closed form solution.	TD-to-position calculations are much simpler than above three; done once. Highly accurate. Predictable error, if present. Modest memory storage requirements. Faster than above three.	Spherical angle to geocentric position calculation constants need to be computed offline. More difficult than with first two to insert propagation corrections.

Table 3-1. Tradeoff Considerations for Various TD-to-Position Methods.

or points where convergence was poor. The results of this test are shown in Figure 3-11, which shows randomly selected points along with their corresponding TDs, which were calculated using a program developed by the Naval Oceanographic Office, and the calculated position determined by the direct algorithm using the program in Appendix II. Two tests were conducted; one using a spherical approximation of the earth and the other using an ellipsoidal approximation of the earth. As can be seen, the ellipsoidal version gives more accurate solutions and the solutions tend to show smooth deviations from the correct values, as shown in Figure 3-12. Also, as evident by standard deviations, the spherical approximation shows sharper deviations from the normal errors expected.

It is quite evident that the ellipsoidal model gives better performance than the spherical model. The two models were run using FORTRAN-IV programs of the type shown in Appendix II on an IBM 370/158. The programs used approximately 1300 bytes of memory and executed in 0.02 seconds.

ORIGINAL PAGE IS
OF POOR QUALITY

Chain ID=9960, Triad=M-Y, M-Z (Seneca, Carolina Beach, Dana)
Range < 50 nm

Control Values			Computer Results			
LAT/ LONG	TDX/ TDY	RNG/ BRG.	LAT/LONG (SPHEROID)	RNG/BRG ERROR	LAT/LONG (ELLIPSOID)	RNG/BRG ERROR
39 0 0	42511.78	35.5	38 59 28.19	38.81	39 0 1.21	1.21
80 0 0	57694.23	77.7	79 59 8.97	51.03	80 0 1.76	1.76
39 30 0	42802.08	39.1	39 29 18.75	41.25	39 29 59.01	0.99
80 30 0	57590.86	16.7	80 29 14.35	45.65	80 29 59.89	0.11
39 0 0	42436.48	36.3	38 59 26.11	33.89	39 0 1.43	1.43
81 30 0	57006.86	282.1	81 29 25.56	34.44	81 29 52.20	7.80
38 0 0	41839.48	53.9	37 59 44.45	15.55	38 0 5.82	5.82
81 0 0	57049.77	193.1	80 59 24.07	35.93	80 59 57.20	2.80

average error 0.75 nm 0.06 nm
standard deviation 0.16 nm 0.05 nm

50 < Range < 150 nm

41 0 0	43700.85	127.8	40 58 51.50	1 8.50	40 59 53.08	6.92
80 30 0	57984.40	4.9	80 29 11.61	48.39	80 30 0.66	0.66
40 0 0	42971.44	89.2	39 59 7.76	52.24	39 59 57.91	2.09
82 0 0	56966.91	319.5	81 59 32.53	27.47	81 59 53.63	6.37
39 0 0	42356.71	152.7	38 59 21.82	38.18	39 0 2.75	2.75
84 0 0	55769.31	273.8	84 0 1.98	1.98	83 59 43.96	16.04
38 0 0	41861.78	118.7	37 59 41.16	18.84	38 0 6.43	6.43
83 0 0	56157.62	244.4	82 59 51.60	8.40	82 59 48.52	11.48
37 0 0	41225.11	113.1	37 0 1.54	1.54	37 0 10.60	10.60
81 0 0	56896.70	186.3	80 59 26.44	33.56	80 59 56.81	3.19
37 30 0	41504.07	101.3	37 29 54.73	5.27	37 30 7.75	7.75
79 30 0	57561.40	144.2	79 29 5.51	54.49	79 30 2.64	2.64
40 0 0	43361.77	144.2	39 59 12.70	47.30	39 59 56.21	3.79
78 0 0	58808.94	61.2	77 58 40.29	1 19.71	78 0 10.05	10.05
38 30 0	42267.50	130.8	38 29 38.25	21.75	38 30 3.19	3.19
78 0 0	58344.52	99.1	77 58 43.81	1 16.19	78 0 8.84	8.84

average error 0.84 nm (spherical) 0.15 nm(ellipsoidal)
standard deviation 0.37 nm (spherical) 0.04 nm(ellipsoidal)

Figure 3-11. Test Results of Explicit Coordinate Converter Program Using Both Spherical and Ellipsoidal Earth Models.

ORIGINAL PAGE IS
OF POOR QUALITY

150 < Range < 300 nm

35 0 0	40711.29	309.6	35 0 25.32	25.32	35 0 20.87	20.87
85 0 0	55394.47	222.6	85 0 27.74	27.74	84 59 39.62	20.38
34 30 0	39907.81	262.5	34 30 43.23	43.23	34 30 22.96	22.96
81 0 0	56645.87	182.8	80 59 30.39	29.61	80 59 56.21	3.79
38 30 0	42436.63	270.7	38 29 36.98	23.02	38 30 3.35	3.35
75 0 0	59149.22	93.0	74 58 6.13	1 53.87	75 0 14.06	14.06
43 0 0	44322.59	253.8	42 58 2.61	1 57.39	42 59 49.84	10.16
82 0 0	57669.90	347.4	81 59 24.07	35.93	81 59 54.84	5.16

average error 1.23 nm(spherical) 0.30 nm(ellipsoidal)
standard deviation 0.66 nm(spherical) 0.13 nm(ellipsoidal)

Figure 3-11. Continued.

ORIGINAL PAGE IS
OF POOR QUALITY

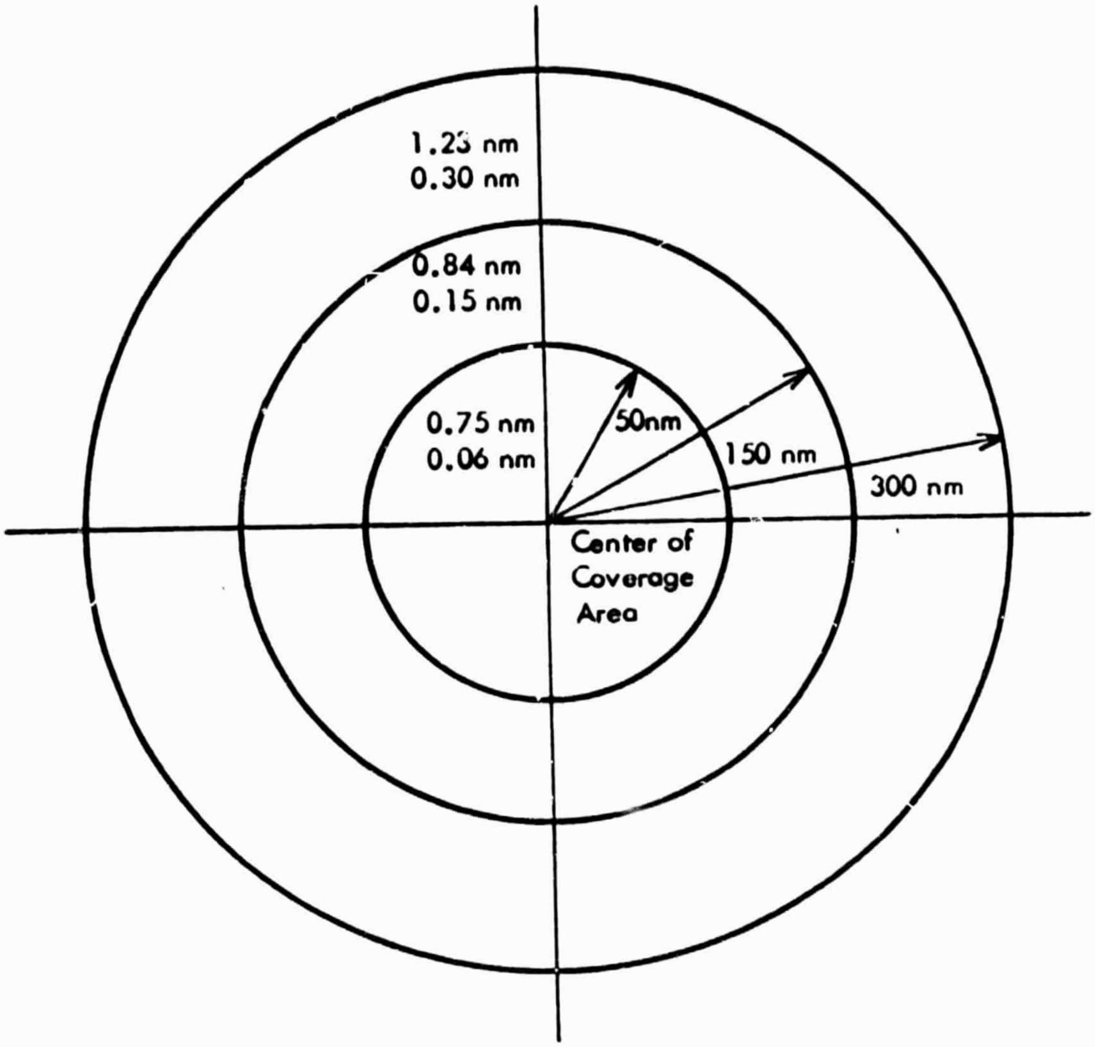

Figure 3-12. General Relationship of Conversion Errors to Distance from Central Point. The top number in each circle is the spherical model error, and the bottom number is the ellipsoidal model error.

IV. THE MICROCOMPUTER SYSTEM

This chapter will detail the process of running the direct TD-to-position program described in the last chapter on a microprocessor-based system. A description of the microprocessor system to be used will be given along with the necessary additional hardware required. Also, special software problems unique to this process will be discussed. Finally, results will be given showing how well the coordinate conversion program works and how well it can be used with a Loran-C software-based sensor processor receiver in a general aviation environment.

Most popular microprocessors on the market contain an accumulation register (accumulator) which is eight bits wide. All data processing, then, takes place on eight-bit words. These processors contain other registers; for instance, most have a 16-bit register which drives the processor's address bus to fetch instructions and data, there are usually several registers internal to the processor which can be used for scratch pad purposes in calculations or as indices to an address, and there is usually a register for indicating the status of the processor at any given time. Most processors are not suited to doing arithmetic work such as multi-precision addition, subtraction, multiplication and division when only eight-bit words can be handled at one time. Because of the range of numbers involved in the calculations in the coordinate converter, it is necessary to use floating-point numbers; arithmetic computations on these can be rather complex.

A microprocessor which has been used previously in Loran-C development work at Ohio University is the MOS Technology MOS6502; the internal architecture of this microprocessor is shown in Figure 4-1. This microprocessor consists of one accumulator where arithmetic and logical results are collected, two eight-bit index registers, one eight-bit stack pointer, a six-bit flag or status register and a program counter [30]. Figure 4-2 is a list of instructions and addressing modes for the 6502 [31]. The software development support for this system includes a symbolic assembler [32] and a simulator both of which reside on the IBM 370/158 at Ohio University. Most of the microprocessor work done at Ohio University was done on a Super Jolt microcomputer, shown schematically in Figure 4-3.

A. <u>Design Considerations</u>. The coordinate converter algorithm described in the last chapter was written in FORTRAN-IV for testing and evaluation. In order to run the coordinate converter on the Super Jolt, it is necessary to convert the FORTRAN-IV program statements to assembly language statements which the 6502 can then run after the statements have been assembled. The main problem to be solved in doing this is the mathematical operations which must be performed. A table is set aside in memory which contains various numbers used as constants and also numbers used for intermediate calculations. As was mentioned, it is necessary to use a floating-point format for repre-

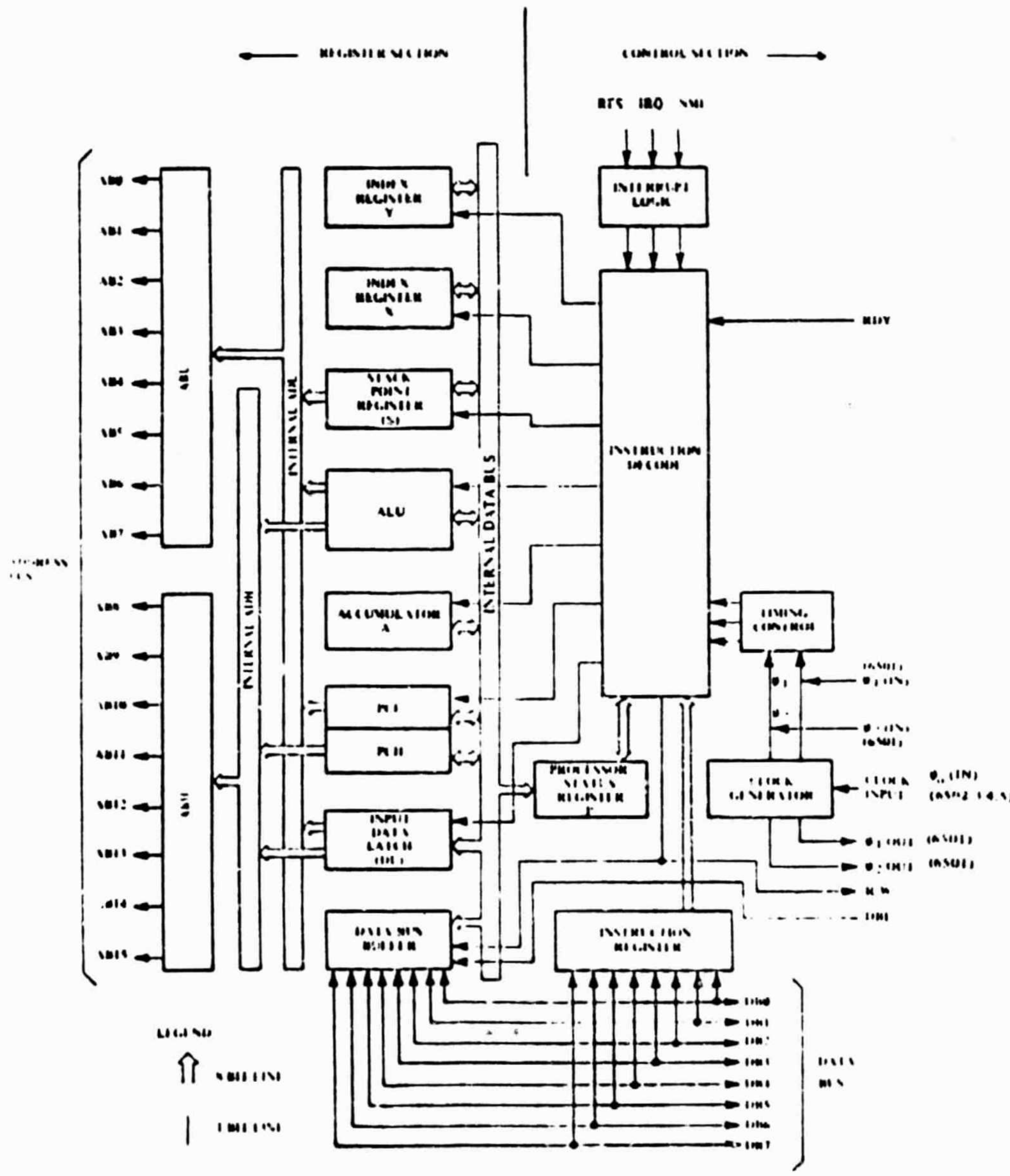

Figura 4-1. Intemal Architecture of MOS6502 Microprocessor.

ORIGINAL PAGE IS
OF POOR QUALITY

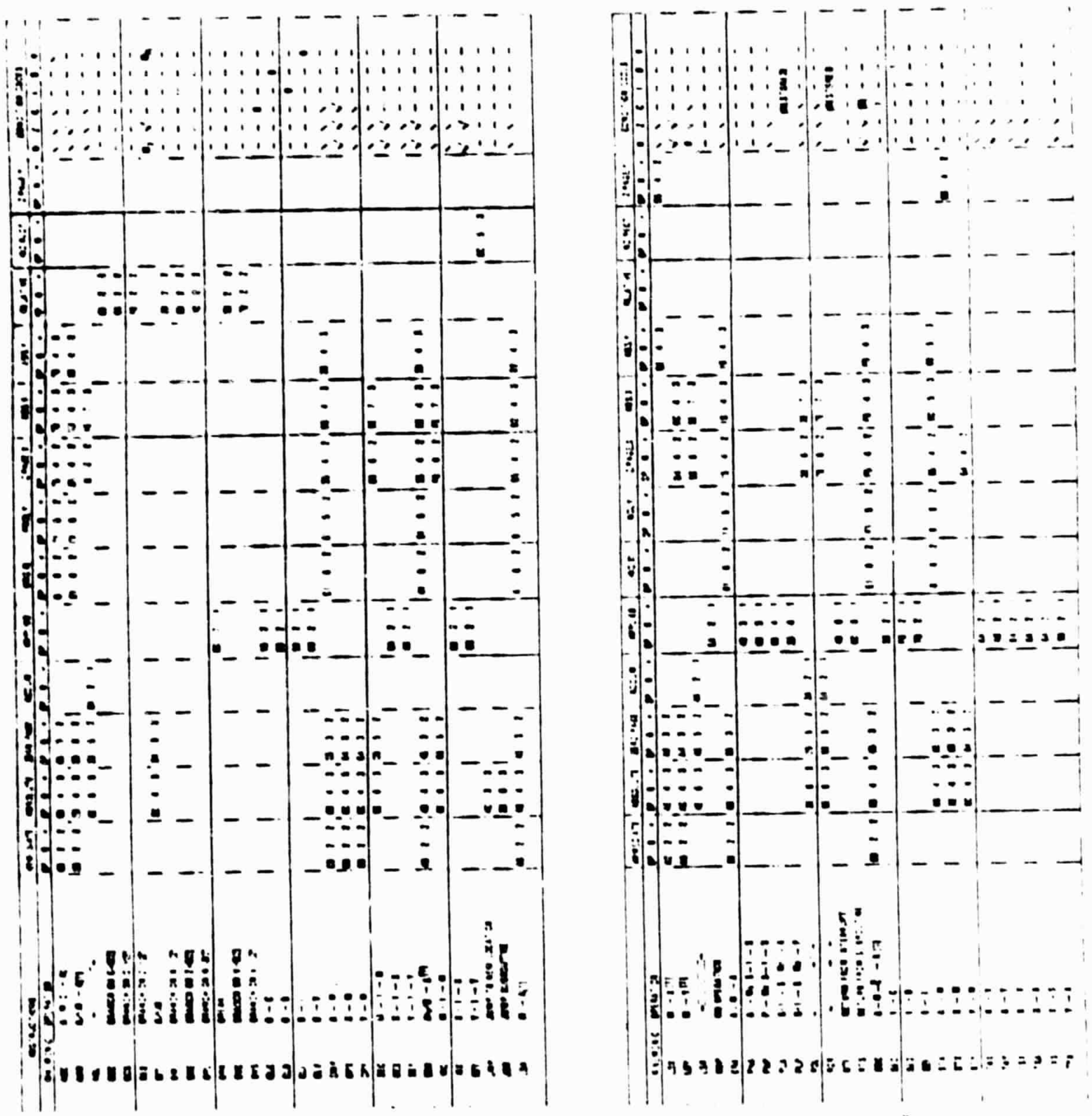

Figure 4-2. Instruction and Addressing Modes of MOS6502.

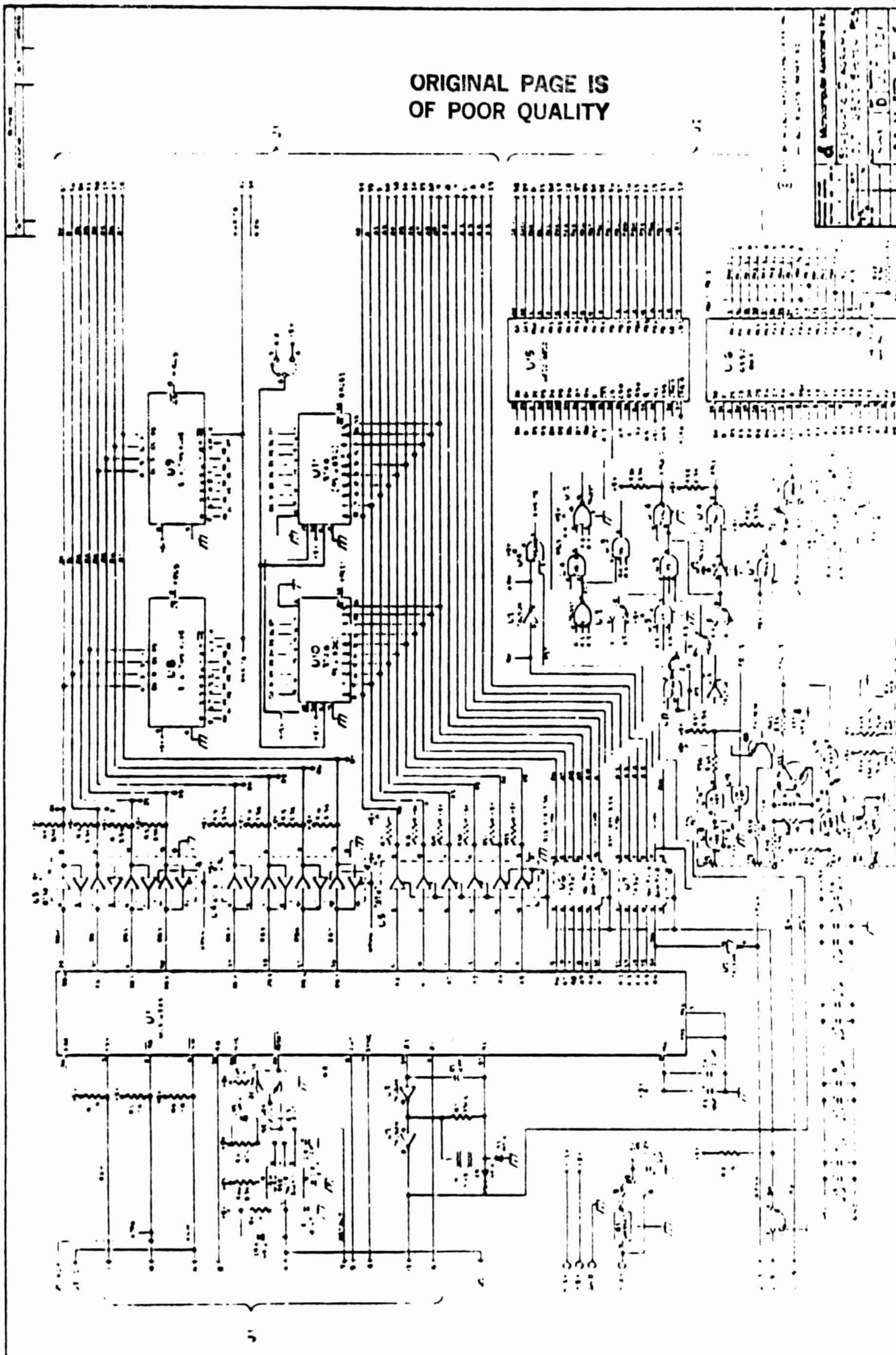

Figure 4-3. Schematic Diagram of Jolt Microcomputer.

septing the numbers since the numerical range is approximately 10^{-7} to 10^{6}, approximately 13 orders of magnitude. The software to do floating-point arithmetic on a microprocessor uses a large amount of memory and can take several milliseconds to do a floating-point multiplication if the difference between the range of numbers is large.

The amount of memory in most small microprocessor systems is limited, especially for applications requiring much arithmetic processing and interfacing to other devices. The microprocessor system used at Ohio University for Loran-C development work has five kilobytes of read/write memory available and provisions for adding more memory, either read/write or permanent type. To cut down size and cost, it is desirable to place the coordinate converter control program in a 2048 byte erasable, read-only memory (EPROM) and use approximately 500 bytes, or less, of read/write memory for scratch pad calculations, storage of computed results, and transfers of data to and from the sensor-processor. This represents a practical requirement from the sense of designing and building such a system.

As part of the project, a peripheral mathematics processor was used to do the necessary floating-point calculations. This device is the Am9511 by Advanced Micro Devices and is shown in Figure 4-4. This device is designed to be used on microprocessor systems which use an eight-bit data bus. It can handle 16-bit and 32-bit fixed point arithmetic and 32-bit floating point arithmetic. Figure 4-5 is a list of instructions for the Am9511A.

The input to the coordinate converter comes from a Loran-C sensor routine which runs in the same microprocessor system [33]. The output of the coordinate converter consists of the computed lat/long for the received TDs and the range and bearing to the selected waypoint. These values are displayed on a video display unit connected to the microprocessor system [34]. These computed values, along with the received TDs are also recorded on a data recording unit which is interfaced to the microprocessor system. The overall scheme of the microcomputer system used for the coordinate converter is shown in Figure 4-6.

B. System for Coordinate Conversion. In order to interface the Am9511A to the Jolt microcomputer system, it was necessary to add additional hardware to the Jolt to handle device selection and data transfer, and also to write a set of subroutines in assembly language to handle the transfer of data in user memory to the 9511 and vice versa, to command the 9511 to perform a function and to check the status of the 9511. Since the 9511 was designed to interface mainly with the 8080, 8085 and Z80 family of microprocessors, which use a different method of handling data transfers and monitoring of peripheral devices, an M6820 peripheral interface adapter (PIA) was used for the hand-shaking. The method of doing this is illustrated in Figure 4-7. The M6820 consists of two eight-bit ports and several other registers

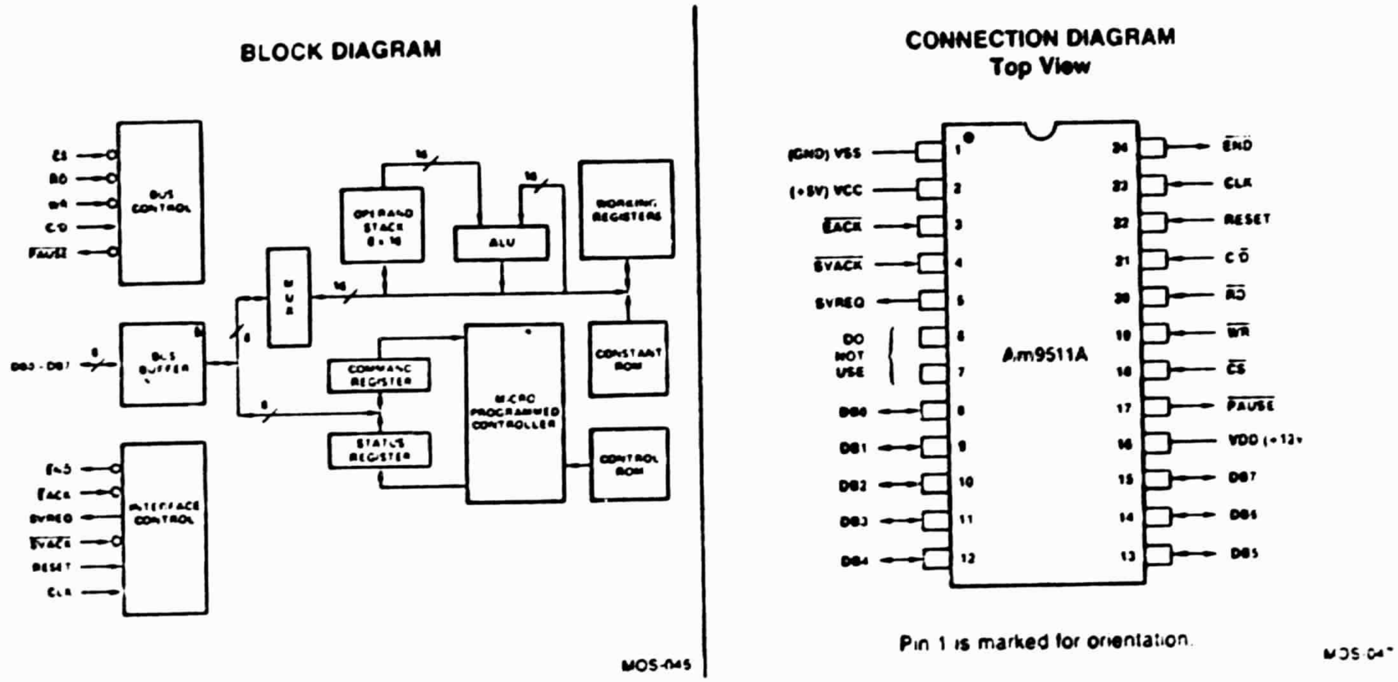

Figure 4-4. Internal Architecture and Connection Diagram of Am9511A Mathematics Processor.

ORIGINAL PAGE IS
OF POOR QUALITY

COMMAND SUMMARY

Command Code 7	6	5	4	3	2	1	0	Command Mnemonic	Command Description
FIXED-POINT 16-BIT									
sr	1	1	0	1	1	0	0	SADD	Add TOS to NOS. Result to NOS. Pop Stack.
sr	1	1	0	1	1	0	1	SSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
sr	1	1	0	1	1	1	0	SMUL	Multiply NOS by TOS. Lower half of result to NOS. Pop Stack.
sr	1	1	1	0	1	1	0	SMUU	Multiply NOS by TOS. Upper half of result to NOS. Pop Stack.
sr	1	1	0	1	1	1	1	SDIV	Divide NOS by TOS. Result to NOS. Pop Stack.
FIXED-POINT 32-BIT									
sr	0	1	0	1	1	0	0	DADD	Add TOS to NOS. Result to NOS. Pop Stack.
sr	0	1	0	1	1	0	1	DSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
sr	0	1	0	1	1	1	0	DMUL	Multiply NOS by TOS. Lower half of result to NOS. Pop Stack.
sr	0	1	1	0	1	1	0	DMUU	Multiply NOS by TOS. Upper half of result to NOS. Pop Stack.
sr	0	1	0	1	1	1	1	DDIV	Divide NOS by TOS. Result to NOS. Pop Stack.
FLOATING-POINT 32-BIT									
sr	0	0	1	0	0	0	0	FADD	Add TOS to NOS. Result to NOS. Pop Stack.
sr	0	0	1	0	0	0	1	FSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
sr	0	0	1	0	0	1	0	FMUL	Multiply NOS by TOS. Result to NOS. Pop Stack.
sr	0	0	1	0	0	1	1	FDIV	Divide NOS by TOS. Result to NOS. Pop Stack.
DERIVED FLOATING-POINT FUNCTIONS									
sr	0	0	0	0	0	0	1	SQRT	Square Root of TOS. Result in TOS.
sr	0	0	0	0	0	1	0	SIN	Sine of TOS. Result in TOS.
sr	0	0	0	0	0	1	1	COS	Cosine of TOS. Result in TOS.
sr	0	0	0	0	1	0	0	TAN	Tangent of TOS. Result in TOS.
sr	0	0	0	0	1	0	1	ASIN	Inverse Sine of TOS. Result in TOS.
sr	0	0	0	0	1	1	0	ACOS	Inverse Cosine of TOS. Result in TOS.
sr	0	0	0	0	1	1	1	ATAN	Inverse Tangent of TOS. Result in TOS.
sr	0	0	0	1	0	0	0	LOG	Common Logarithm (base 10) of TOS. Result in TOS.
sr	0	0	0	1	0	0	1	LN	Natural Logarithm (base e) of TOS. Result in TOS.
sr	0	0	0	1	0	1	0	EXP	Exponential (e^x) of TOS. Result in TOS.
sr	0	0	0	1	0	1	1	PWR	NOS raised to the power in TOS. Result in NOS. Pop Stack.
DATA MANIPULATION COMMANDS									
sr	0	0	0	0	0	0	0	NOP	No Operation
sr	0	0	1	1	1	1	1	FIXS	Convert TOS from floating point to 16 bit fixed point format.
sr	0	0	1	1	1	1	0	FIXD	Convert TOS from floating point to 32 bit fixed point format.
sr	0	0	1	1	1	0	1	FLTS	Convert TOS from 16 bit fixed point to floating point format.
sr	0	0	1	1	1	0	0	FLTD	Convert TOS from 32 bit fixed point to floating point format.
sr	1	1	1	0	1	0	0	CHSS	Change sign of 16 bit fixed point operand on TOS.
sr	0	1	1	0	1	0	0	CHSD	Change sign of 32 bit fixed point operand on TOS.
sr	0	0	1	0	1	0	1	CHSF	Change sign of floating point operand on TOS.
sr	1	1	1	0	1	1	1	PTOS	Push 16 bit fixed point operand on TOS to NOS (Copy).
sr	0	1	1	0	1	1	1	PTOD	Push 32 bit fixed point operand on TOS to NOS (Copy).
sr	0	0	1	0	1	1	1	PTOF	Push floating point operand on TOS to NOS. (Copy)
sr	1	1	1	1	0	0	0	POPS	Pop 16 bit fixed point operand from TOS. NOS becomes TOS.
sr	0	1	1	1	0	0	0	POPD	Pop 32 bit fixed point operand from TOS. NOS becomes TOS.
sr	0	0	1	1	0	0	0	POPF	Pop floating point operand from TOS. NOS becomes TOS.
sr	1	1	1	1	0	0	1	XCHS	Exchange 16 bit fixed point operands TOS and NOS.
sr	0	1	1	1	0	0	1	XCHD	Exchange 32 bit fixed point operands TOS and NOS.
sr	0	0	1	1	0	0	1	XCHF	Exchange floating point operands TOS and NOS.
sr	0	0	1	1	0	1	0	PUPI	Push floating point constant π onto TOS. Previous TOS becomes NOS.

NOTES

1. TOS means Top of Stack. NOS means Next on Stack.
2. AMD Application Brief "Algorithm Details for the Am9511A APU" provides detailed descriptions of each command function, including data ranges, accuracies, stack configurations, etc.
3. Many commands destroy one stack location (bottom of stack) during development of the result. The derived functions may destroy several stack locations. See Application Brief for details.
4. The trigonometric functions handle angles in radians, not degrees.
5. No remainder is available for the fixed point divide functions.
6. Results will be undefined for any combination of command coding bits not specified in the table.

Figure 4-5. Instruction Set of Am9511A.

ORIGINAL PAGE IS
OF POOR QUALITY

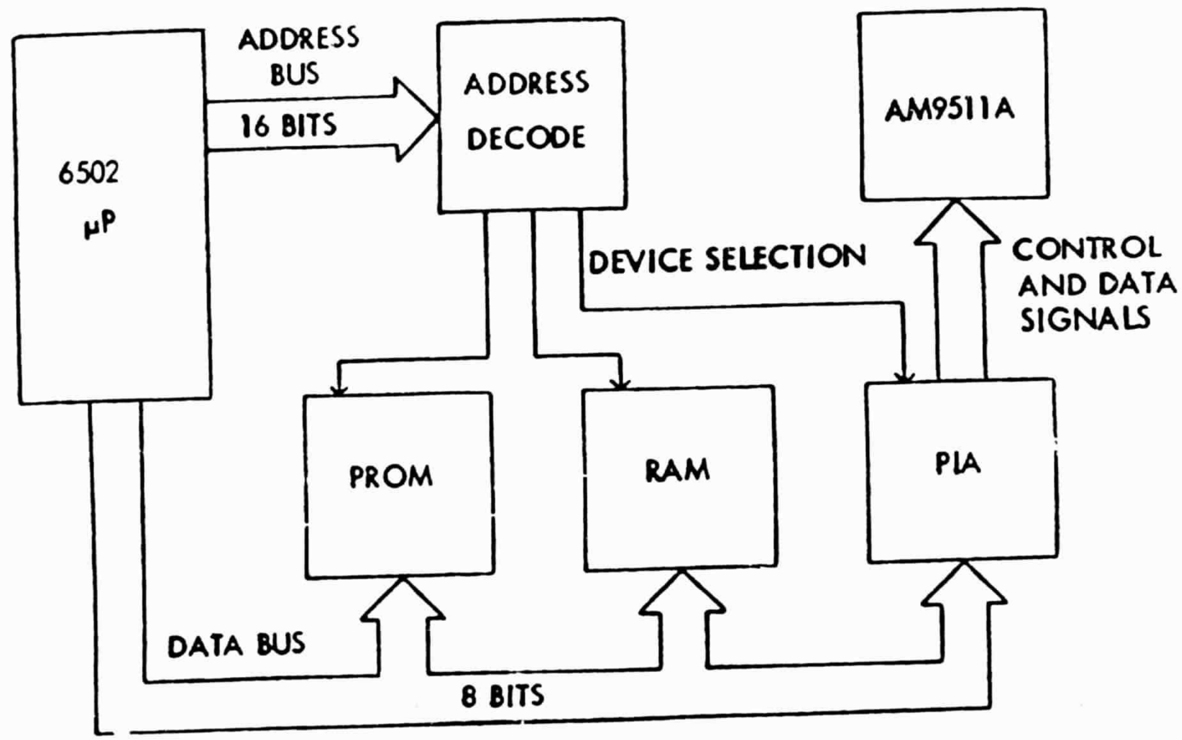

Figure 4-6. Block Diagram of Microcomputer Coordinate Converter.

ORIGINAL PAGE IS
OF POOR QUALITY

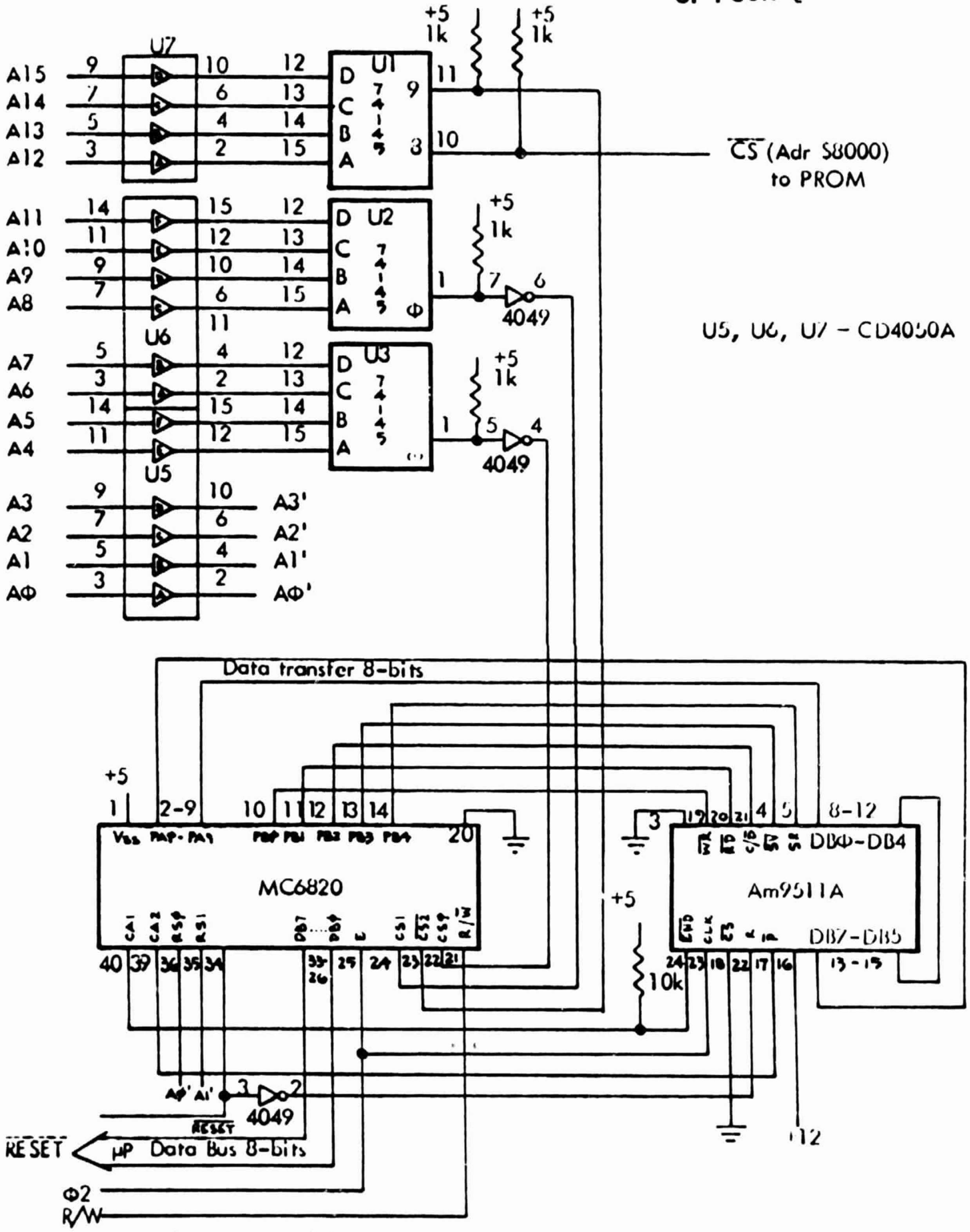

Figure 4-7. Schematic Diagram of Microcomputer Interface to 9511
Designed for TD-to-position Computation System.

used to interface to peripheral devices as shown in Figure 4-8 [35]. Side A of the PIA is used to transfer eight bits of data between the 9511 and the microprocessor system. Various ports on side B are used to set and read I/O lines on the 9511 for reading data, writing data, selecting internal registers in the 9511, etc. The interrupt registers on the M6820 are used by the 9511 to indicate when the 9511 has made valid data available to the microprocessor and when the 9511 has completed execution of a command.

The next part of the interfacing to the 9511 is the development of a standard set of programs (subroutines) which are used to initialize the hardware interface and the 9511, to write, or load a single floating-point (32-bit) number, to read or unload a floating-point (32-bit) number, to send an eight-bit word to the 9511 representing a command to be executed, and to read the 9511 eight-bit status register. Figure 4-9 gives a summary of the necessary procedures to do the items listed above. It is desired to make these subroutines complete to the point where the main program can specify a pointer to a particular number or specify a particular operation to be executed and then call the appropriate subroutine to do the function and return when the operation is completed. To this end, an area of storage in read/write memory is reserved for storing all numbers which will eventually be passed to and from the 9511. The base address (beginning of table address) is stored at an address to which the subroutines have common access, then all the numbers are referenced by a single eight-bit offset from the base address. Since each number occupies 32 bits, or four bytes, in memory, 64 numbers can be stored by this procedure. If more than 64 numbers must be stored, then a second number table and a different base address must be set up.

Figure 4-10 shows a set of flow diagrams to illustrate the logic flow of a set of standard subroutines to do the processes previously discussed. The first, called "PINT," initializes the M6820 PIA, which consists of setting the data ports to either inputs or outputs, as required, and setting up the interrupt registers, then the control lines to the 9511 are deselected so that the 9511 will not respond to invalid data during the initialization period. Also included in this subroutine is a section which moves a set constants stored in the permanent memory to the read/write number table and initializes an area of memory used for video display work.

Subroutine "PUSH" is used to copy a four-byte number from read/write memory onto the stack of the 9511. The Y-register contains the eight-bit offset from the base address of the number table for the number to be copied. In this case, four individual transfers are made, since the number is 32-bits long and only eight bits are transferred at a time. The number which is copied appears at the top of the stack in the 9511 and any other numbers stored on the stack are moved further down in the stack by one location. The number stored at the bottom of stack is lost.

ORIGINAL PAGE IS
OF POOR QUALITY

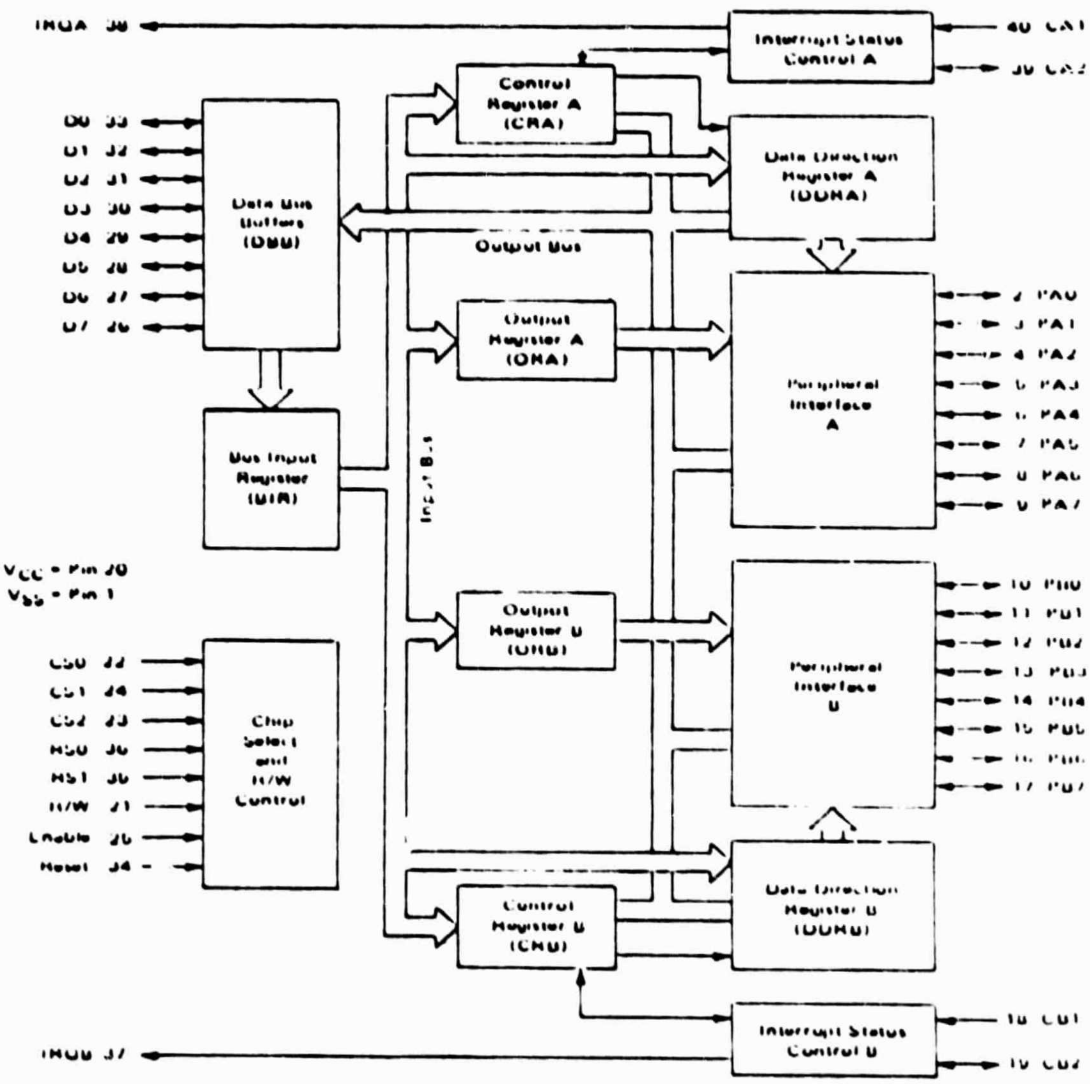

Figure 4-8. Internal Architecture of M6820 PIA
From Motorola Specifications.

Operand entry

1. Place least significant byte on data bus.
2. Set C/$\overline{D}$ low.
3. Set $\overline{CS}$ low.
4. Set $\overline{WR}$ low. Pause will become low.
5. Pause returns high.
6. Set $\overline{WR}$ high-data, C/$\overline{D}$, $\overline{CS}$ may then change.
7. Repeat above six to write entire 32-bit number.

Data removal

1. Set C/$\overline{D}$ low.
2. Set $\overline{CS}$ low.
3. Set $\overline{RD}$ low. Pause becomes low.
4. Pause returns high - data is then available, most significant byte first.
5. Set $\overline{RD}$ high - $\overline{CS}$, R/$\overline{D}$, data may then change.
6. Repeat above five to read entire 32-bit number.

Command initiation

1. Place command byte on data bus.
2. Set C/$\overline{D}$ high.
3. Execute steps 3-6 of operand entry.
4. $\overline{END}$ output remains low until command completes.

Status read

1. Set C/$\overline{D}$ high.
2. Execute steps 2-5 of data removal.
3. Status byte is available on bus.

Figure 4-9. Summary of Steps to Communicate to 9511.

ORIGINAL PAGE IS
OF POOR QUALITY

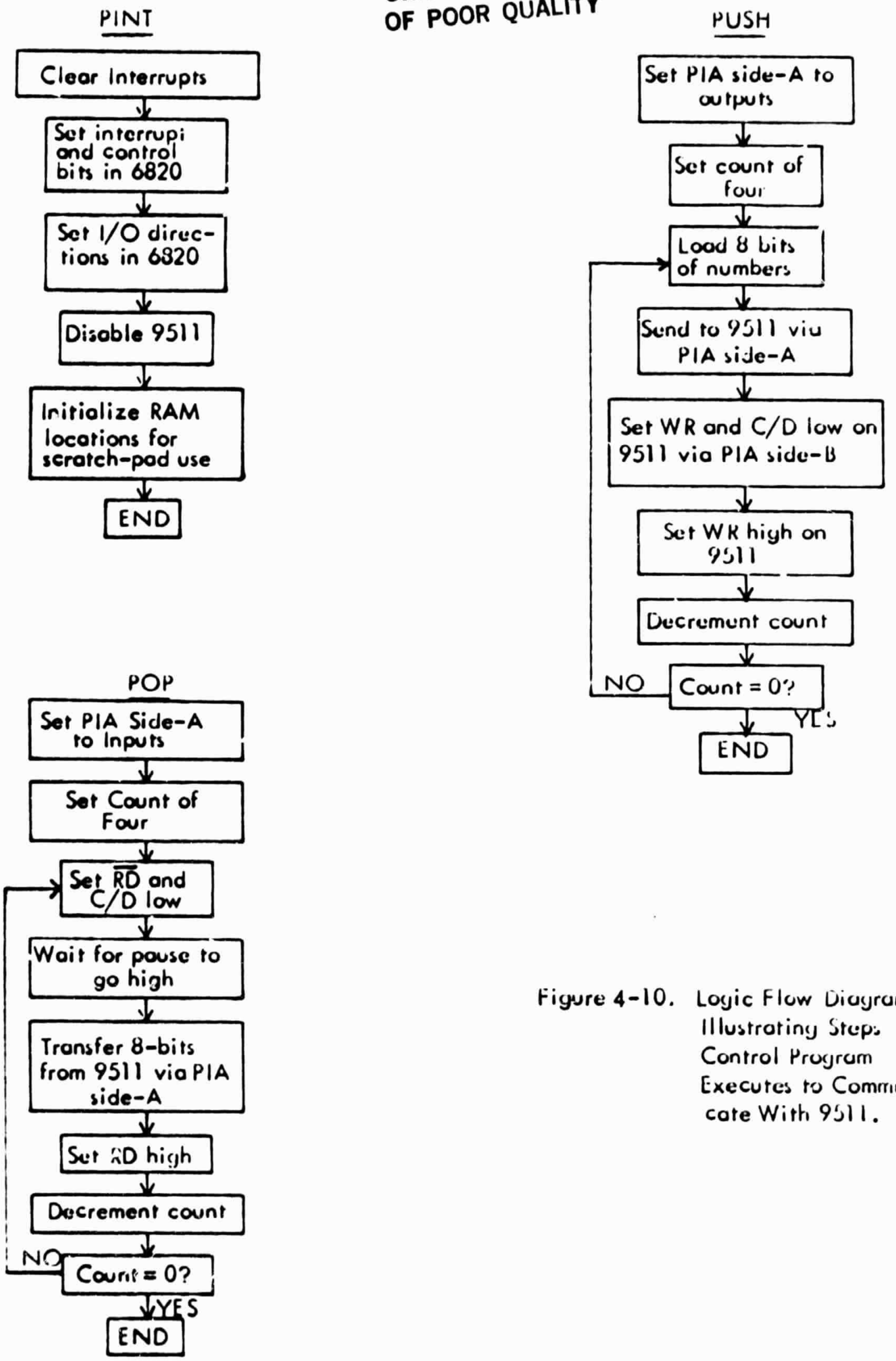

Figure 4-10. Logic Flow Diagrams Illustrating Steps Control Program Executes to Communicate With 9511.

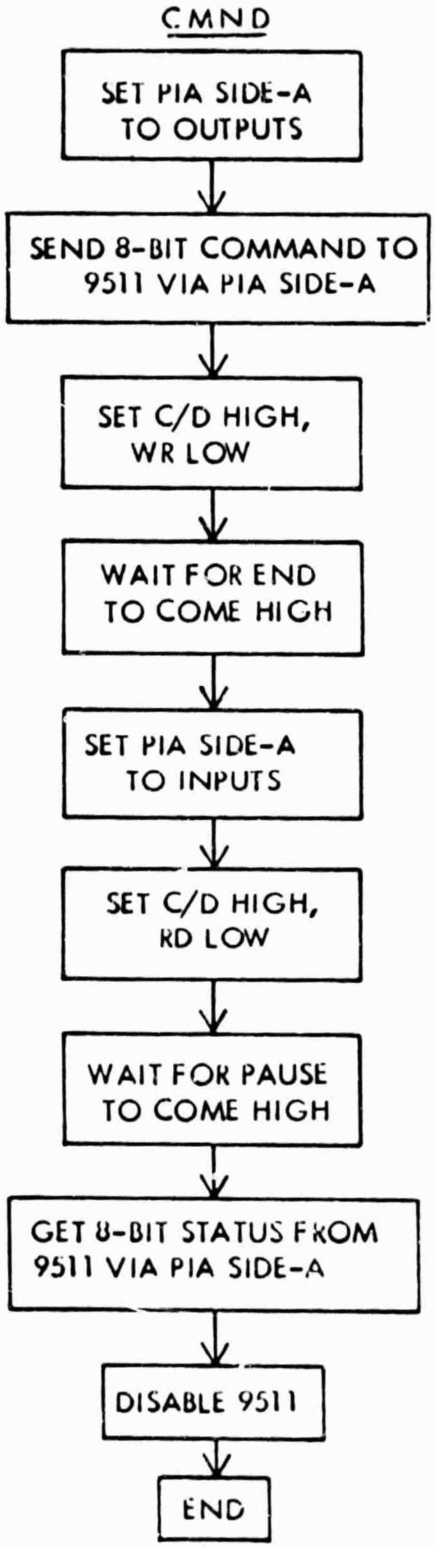

Figure 4-10. Continued.

Subroutine "POP" is used to do the opposite of the above; a 32-bit number is removed from the top of stack in the 9511 and placed in storage. The address where the number is stored is pointed to by the Y-register offset relative to the base address. After this operation is complete, the number that was at the top of stack is lost and any other numbers in the stack are incremented up by one position. This subroutine incorporates a test of the pause output of the 9511 before loading data over the data bus. In some instances, it may take the 9511 several microseconds to present the data; when the data is available, an interrupt bit in the PIA is set. This is not necessary for writing data, however.

Subroutine "CMND" is used to command the 9511 to perform a given function, the function code is contained in the accumulator. After the 9511 has accepted the command, it immediately begins execution of that command. During this time, its status register will indicate that it is busy; when the command has finished execution, the status line will set an interrupt bit in the PIA. The subroutine loops until this bit is set before returning to the main program; thus, on return, the function will be completed and the main program need not check for completion.

As part of the "CMND" subroutine, a check is made of the status register to determine the final outcome of the completed command. This may consist of an error in processing because of division by zero, an input number out of range, etc. The return code is returned in the accumulator for further processing by the main program.

Other subroutines are necessary for the conversion of binary coded decimal (BCD) input data representing the time differences to the internal floating-point format used by the 9511 and from floating-point to BCD for displaying the final results. One of the commands is available with the 9511 is one for converting a fixed-point (integer) number to floating-point. The fixed-point number must be in binary and it is not possible to represent a fraction using this command. The time differences from the receiver are in base 10 in units of microseconds with a single digit representing a fraction of a microsecond, for example, 42594.2. The two time differences are stored at a certain location in memory by the receiver and are in packed format; i.e., two digits occupy one byte of memory. Before the TD-to-position calculations may be performed, these two TDs must be converted to binary, then to floating point. In the microcomputer program, this is done by treating each TD without regard to the the decimal point; the actual binary TD is multiplied by 10, the correct TD is obtained by multiplying by 10^{-7} to get them in microseconds.

The process of converting a base 10 number to binary is illustrated in Figure 4-11. This is done by starting with the most significant digit (MSD) of the base 10 number and adding it to a register then performing a binary multiplication by 10. In this way, each digit position of the base 10 number is converted to binary by

ORIGINAL PAGE IS
OF POOR QUALITY

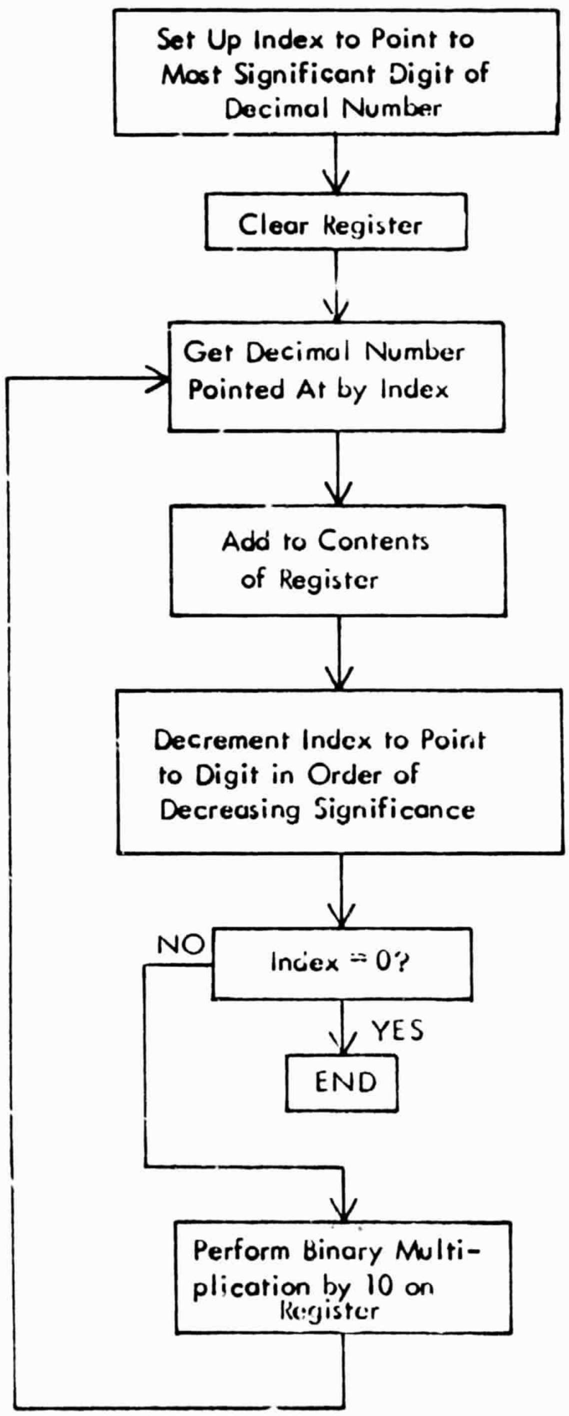

Figure 4-11. Method of Converting Decimal (BCD) Number to Binary and Example.

ORIGINAL PAGE IS
OF POOR QUALITY

Example

BCD number - 1981
register contents 0

1. get MSD - 1

2. add to register - 1

3. multiply register by 10 - A (hex)

4. add next digit - 13 (hex)

5. multiply by 10 - BE (hex)

6. add next digit - C6 (hex)

7. multiply by 10 - 7BC (hex)

8. add LSD - 7 BD (hex)

9. stop - answer = 7BD (hex)

Figure 4-11. Continued

the multiplication and added to a partial sum to form the final answer.

The binary multiplication is performed by a process called Booth's algorithm [36]. This is also applicable for binary division, as used at the end of the microcomputer coordinate converter program. In this algorithm, the multiplicand is stored in successive locations in memory and the multiplier is stored in the accumulator. The multiplier (in binary) represents a string of zeros and ones; the number of iterations through the multiplication loop is set by the length of the accumulator, in this case, eight. With each iteration, a residue table (initially set to zero) is rotated left, the bit shifted out of the left, reenters on the right. Also, the bits in the accumulator are shifted left, when a one is shifted out, the multiplicand is added to the residue, if a zero is shifted out, no adding occurs. This is illustrated in Figure 4-12. This process is very similar to the way multiplication would be handled with decimal numbers; as the residue is shifted left, it is effectively increased by a power of two. When a one is encountered, the multiplicand is then added to the partial sum.

When the coordinate conversion process is completed, the resulting latitude and longitude of present position and the range and bearing to the waypoint are in the binary floating-point format and must then be converted to BCD for display and use by the Loran-C sensor program if they are to be stored on tape or processed further. For the latitude and longitude, the coordinates must be converted first from radians to degrees, then broken down to the standard degrees, minutes, seconds format. This is done by taking the integer part of the coordinate and storing it as the degree part, then subtracting the integer from the original coordinate and multiplying by 60 to obtain the minutes part. Doing this a second time yields the seconds part. Most of this work is done by using the float-to fix and fix-to-float commands in the 9511.

The binary to BCD conversion is done by taking the binary number and performing a binary division by ten. This yields a quotient and a remainder, the remainder is the corresponding decimal digit of the new base 10 number, working from LSD to MSD. The quotient is then divided by 10 again to yield the next decimal digit. This is continued for a number of iterations corresponding to the number of bits in the original base 2 number. The subroutines which perform these functions appear near the end of the program listing in Appendix IV.

The Loran-C coordinate converter program was designed to run on the Super Jolt microcomputer along with another control program which operated software aided loops that lock onto the Loran-C pulses and compute the corresponding time differences. The executive program in this also operates interfacing to a video display and a data recorder. Certain addresses in page zero (0000 hex to 00FF hex) of the MCS6502

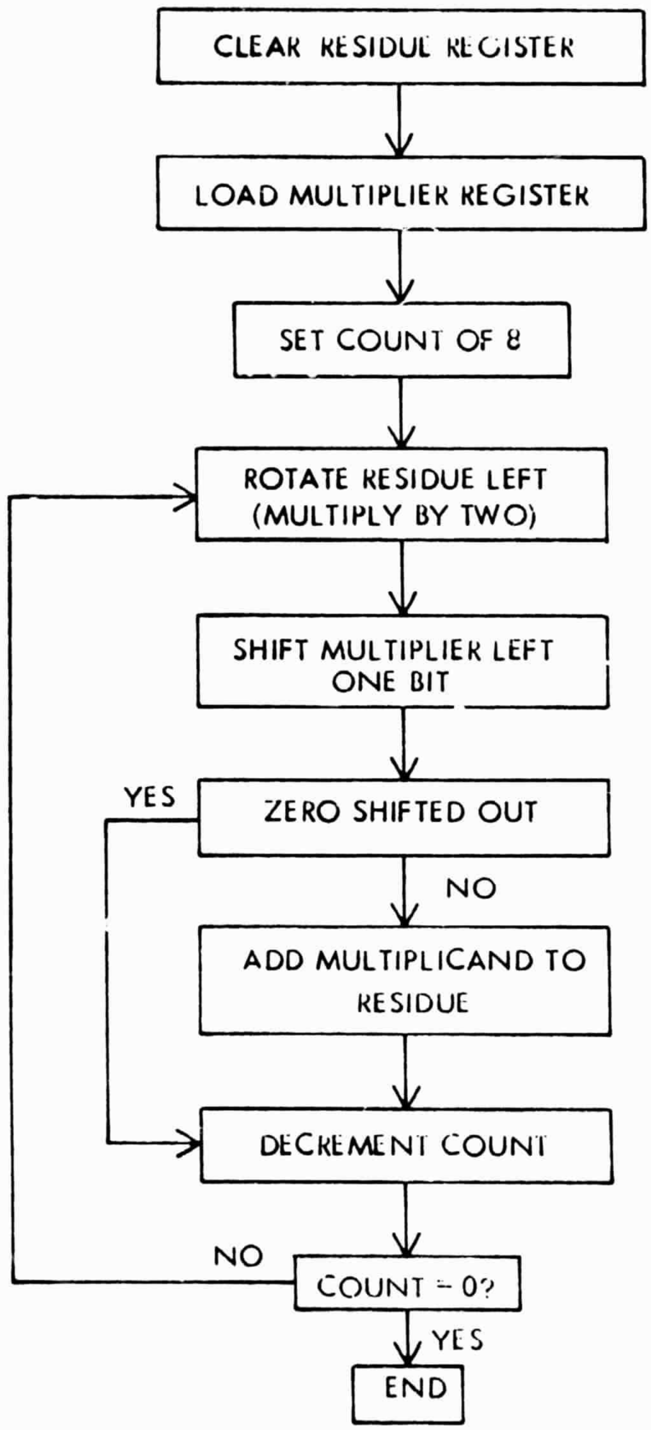

Figure 4-12. Process of Binary Multiplication.

memory scheme are reserved for data communication between the Loran-C sensor program and the coordinate converter program. These are indicated on the first few pages of the program listing in Appendix IV. These locations are used to pass the two time differences to the coordinate converter and to receive the subsequent latitude, longitude, range and bearing. All of these are stored as packed BCD numbers.

The complete microprocessor program for the coordinate conversion is shown in Appendix V and occupies 2045 bytes of EPROM. The program also uses one-half kilobytes of read/write memory for scratch pad calculations. Figure 4-13 shows a photograph of the control hardware including the Am9511, the 6820 PIA, a spare EPROM and associated interfacing components. This board plugs into the Loran-C developmental receiver, shown in Figure 4-14. This unit was tested both on the bench and in several flight checks, which are reported in the next chapter.

As an aid to preparing the coordinate converter software, a service program was written for use on the IBM 370 to allow base 10 numbers to be typed in from the user's terminal, and then converted to the equivalent binary floating-point number for the 9511 and displayed on the user's terminal. The program listing for this appears in Appendix VI.

ORIGINAL PAGE IS
OF POOR QUALITY

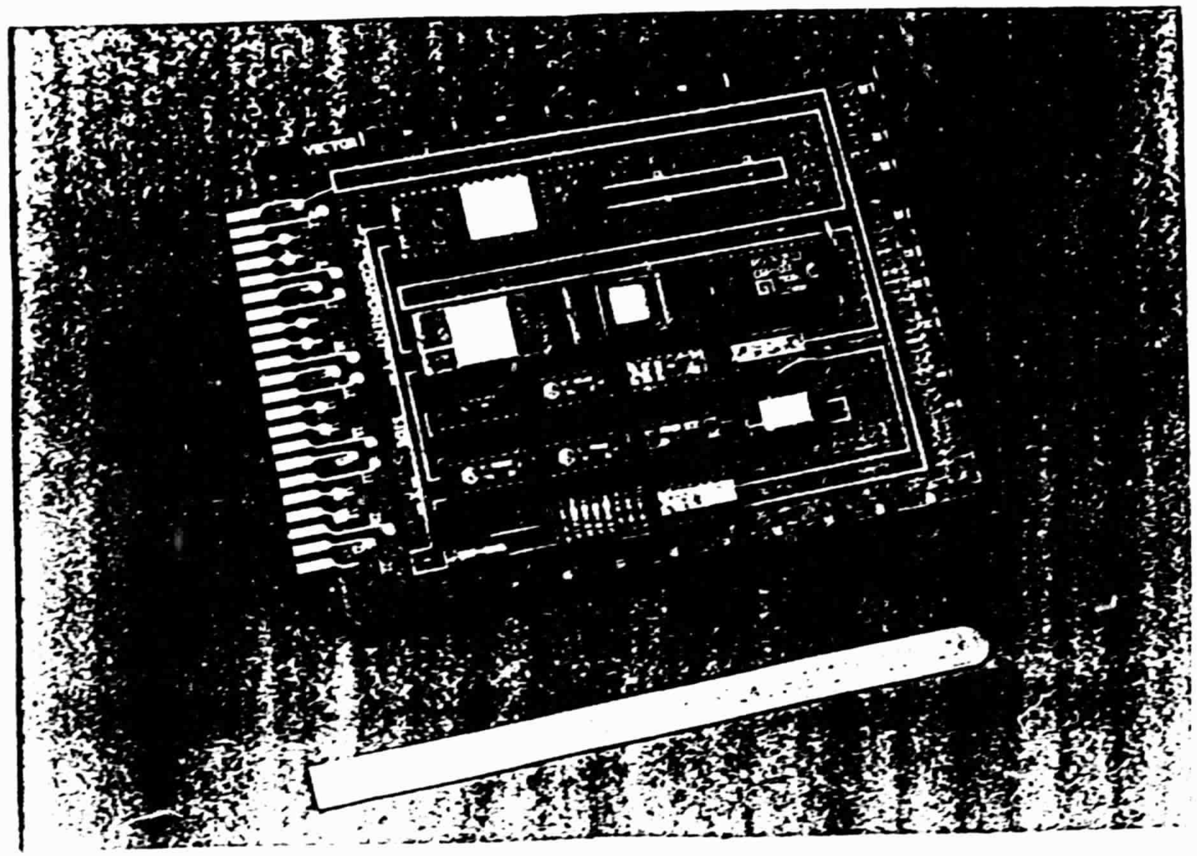

Figure 4-13. Mathematics Processor Board. M6820 is the 40-pin IC; the 24-pin ICs are, from left, the 9511, a 2716 EPROM containing the control program, and a spare EPROM. The other ICs are interface gates.

ORIGINAL PAGE IS
OF POOR QUALITY

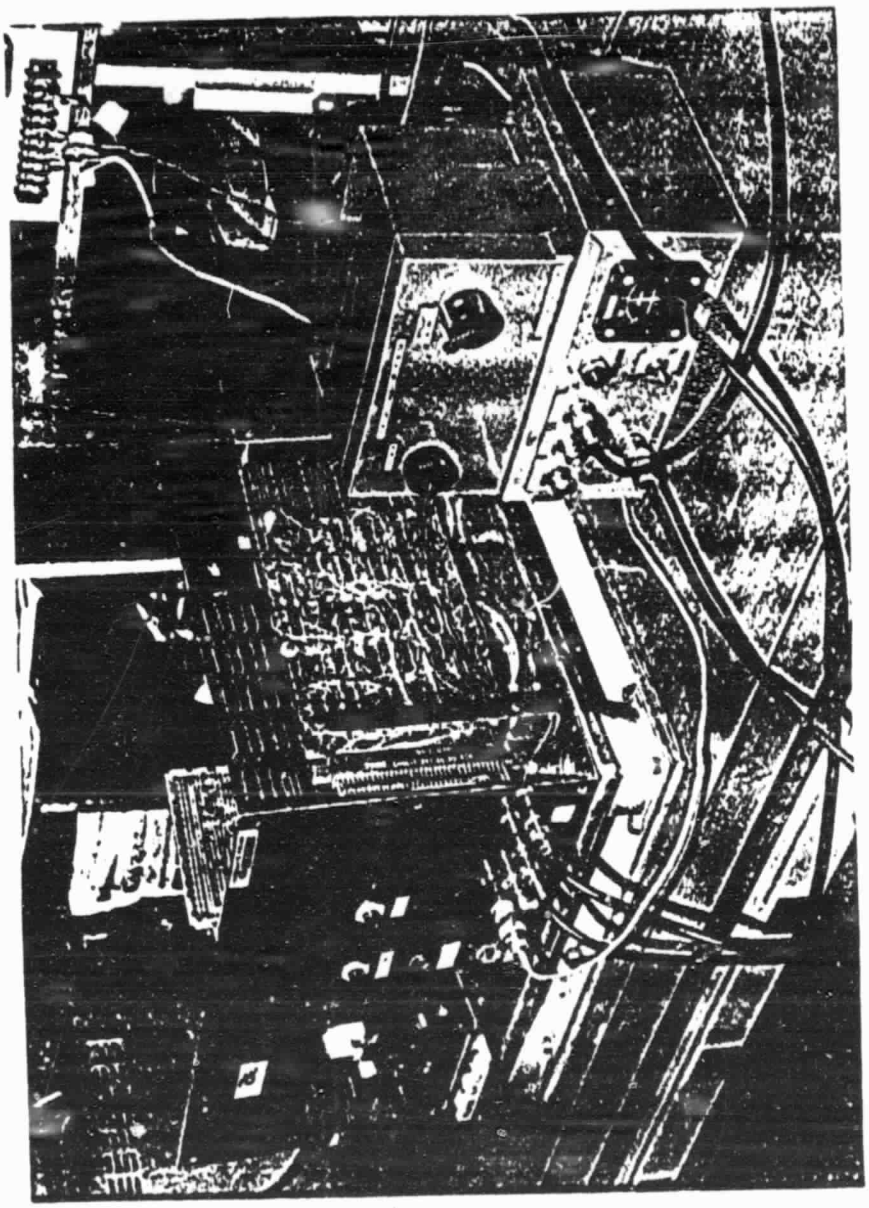

Figure 4-14. Photograph of Coordinate Converter Test Package. The large box contains the Jolt microcomputer, the mathematics board, the Loran-C sensor board, the video interface board, and a digital data recorder. The other two boxes are the Loran-C front-end processors. These were developed at Ohio University under NASA Grant NGR 36-009-017.

ORIGINAL PAGE IS OF POOR QUALITY

V. TESTS ON MICROCOMPUTER SYSTEM

Details of the development of the Loran-C coordinate converter and the microprocessor system have been given. Following is a verification of the performance capabilities of the coordinate converter system. This consists of tests run to determine the accuracy of the process and the suitability of the system to used in an aircraft environment for navigation.

The microcomputer coordinate converter was placed on one 2716 erasable, programmable, read-only memory (EPROM) chip. This EPROM has capacity for 2048 bytes of data; the coordinate converter occupies 2045 bytes. With the Am9511A running at a clock speed of 1 MHz, a complete conversion requires 0.18 second to complete. (The maximum clock speed on the 9511 is 2 MHz and data transfer between the 9511 and the central processor is asynchronous, thus computation throughput could be doubled by running the 9511 at 2 MHz while keeping the 6502 central processor at 1 MHz.) The total memory requirements for the coordinate converter are: 2045 bytes of permanent (EPROM) storage for the control program and constant numbers, 37 bytes of page-zero storage for temporary variables and flags and communication to the Loran-C sensor program, and 120 bytes of general read/write storage for results of intermediate calculations.

Testing of the coordinate converter consisted of checking repeatability, ability to compute correct position given the time differences of a known point, and a check for any singularities or tendencies to converge on the wrong result. Checking of repeatability consisted of using the same time difference readings as input and determining if the coordinate converter gave the same answer each time. In all cases, given a particular TD reading, the coordinate converter yielded the same answer regardless of when the test was done. Thus, the evidence indicates casual environmental factors had no influence on the consistency or computational time of a solution.

A. <u>Testing With Pre-Determined Points</u>. Accuracy checks were accomplished by picking random locations in the Loran-C coverage region for a given triad and computing the corresponding TD readings for that triad of stations. These TDs were then used as input to the coordinate converter and the results were compared with the original position coordinates. These are shown in Figures 5-1 and 5-2. Figure 5-1 are the results obtained using an early version of the program which employed a spherical earth model, while Figure 5-2 shows the results of the latest version which uses an ellipsoidal earth model. In both cases, the center point used in the TD-to-position calculations was chosen to be the center of the Loran-C triad used. This is done by finding the average latitude and longitude of the master and the two selected secondary stations used. Comparing the two results shows that with the spherical approximation, errors tend to increase sharply as the distance from the center point increases beyond 150 nautical miles. In the elliptical case, the errors are much smaller

TDs Y-Z	Lat. (N) / Long. (W) Actual	Measured	Rng (nm.) / Brng (degs.) Actual	Measured
42416.1	39 0 0	39 0 20	19.9	19.6
56765.7	82 0 0	81 59 50	346.6	345.9
42891.9	40 0 0	40 0 9	58.2	58.3
56446.5	83 0 0	83 0 4	134.0	134.4
42971.4	40 0 0	40 0 6	40.8	41.0
56966.9	82 0 0	81 59 53	186.5	186.6
42382.2	39 0 0	39 0 22	46.3	46.0
56272.6	83 0 0	83 0 0	64.9	65.6
42162.8	38 30 0	38 30 23	89.7	89.6
57362.0	80 30 0	80 29 33	303.9	303.1
41924.7	38 0 0	38 0 51	199.7	199.1
54840.5	86 0 0	86 0 11	65.3	66.8
40978.1	36 0 0	36 1 26	218.7	217.6
55686.7	84 0 0	84 0 0	23.8	24.5
43947.6	42 0 0	41 59 52	160.6	160.6
57449.3	82 0 0	82 0 1	181.6	181.6
40294.2	35 0 0	35 1 42	259.1	257.8
56354.4	82 0 0	81 59 46	359.0	358.9
41150.5	37 0 0	37 0 45	202.2	201.9
57642.7	79 0 0	78 59 17	314.5	313.3

STATISTICS

Lat.: mean error = 35.6", stan. dev = 35.5", max = 1' 42"
Long: mean error = -6.5", stan. dev. = 16.1", max. = 43"

Range: mean error = -0.37 nm., stan. dev. = 0.5 nm., max. = 1.3 nm.
Bearing: mean error = 0.06 , stan. dev. = 0.81 , max. = 1.5

Waypoint: Clippinger Labs; 39 19 22.8 N, 82 5 58.4 W

Figure 5-1. Test of Microcomputer Coordinate Converter with Spherical Model.

ORIGINAL PAGE IS
OF POOR QUALITY

Chain ID=9960, Triad=M-Y, M-Z (Seneca, Carolina Beach, Dana)
Range < 50 nm

Control Values			Calculated Values		
LAT/ LONG	TDX/ TDY	RNG/ BRG	LAT/LONG	RNG/ BRG	RNG/BRG ERROR
39 0 0	42511.78	35.5	39 00 01	35.4	0.1
80 0 0	57694.23	257.7	80 00 01	257.8	0.1
39 30 0	42802.08	39.1	39 29 59	39.1	0.0
80 30 0	57590.86	196.7	80 29 59	196.8	0.1
39 0 0	42436.48	36.3	39 00 01	36.1	0.2
81 30 0	57006.86	102.1	81 29 51	101.9	0.2
38 0 0	41839.48	53.9	38 00 06	53.8	0.1
81 0 0	57049.77	13.1	80 59 56	12.9	0.2

average error 0.1 nm, 0.15 degree
standard deviation 0.08 nm, 0.06 degree

50 < Range < 150 nm

41 0 0	43700.85	127.8	40 59 53	127.8	0.0
80 30 0	57984.40	184.9	80 30 00	184.9	0.0
40 0 0	42971.44	89.2	39 59 57	89.1	0.1
82 0 0	56966.91	139.5	81 59 53	139.1	0.4
39 0 0	42356.71	152.7	39 00 03	152.1	0.6
84 0 0	55769.31	93.8	83 59 43	92.8	1.0
38 0 0	41861.78	118.7	38 00 06	118.3	0.4
83 0 0	56157.62	64.4	82 59 48	63.6	0.8
37 0 0	41225.11	113.1	37 00 11	113.1	0.0
81 0 0	56896.70	6.3	80 59 56	6.1	0.2
37 30 0	41504.07	101.3	37 30 08	101.1	0.2
79 30 0	57561.40	324.2	79 30 02	324.6	0.4
40 0 0	43361.77	144.2	39 59 56	143.8	0.4
78 0 0	58808.94	241.2	78 00 10	242.0	0.8
38 30 0	42267.50	130.8	38 30 03	130.3	0.5
78 0 0	58344.52	279.1	78 00 08	279.9	0.8

average error 0.28 nm, 0.55 degree
standard deviation 0.23 nm, 0.35 degree

Figure 5-2. Test Results of Microcomputer Coordinate Converter with Ellipsoidal Model.

150 < Range < 300 nm

35	0	0	40711.29	309.6	35	00	21	309.1	0.5
85	0	0	55394.47	42.6	84	59	38	41.3	1.3
34	30	0	39907.81	262.5	34	30	23	262.6	0.1
81	0	0	56645.87	2.8	80	59	55	2.7	0.1
38	30	0	42436.63	270.7	38	30	03	269.8	0.9
75	0	0	59149.22	273.0	75	00	13	274.8	1.8
43	0	0	44322.59	253.8	42	59	50	253.8	0.0
82	0	0	57669.90	167.4	81	59	54	167.0	0.4

average error 0.38 nm, 0.9 degree
standard deviation 0.41 nm, 0.79 degree

Figure 5-2. Continued.

out to the edge of the coverage region and the results tend to be more stable as a function of distance from the center of the coverage region.

The results presented here are much the same in terms of accuracy as the tests done in Chapter III using the FORTRAN-IV test program. The results in Figures 5-1 and 5-2 also include the range and bearing angle calculations presented in Chapter III.D. The range and bearing calculations allow only a single waypoint to be stored in memory, which must be established before TD-to-position updates are run. There are 11 steps required for each range/bearing computation and this is normally done immediately after the lat/long calculations. The output of the microcomputer program consists of the computed lat/long and the range/bearing stored in memory in both the binary floating-point format and in BCD for data storage and display.

Additional testing of the microcomputer Loran-C system described in Chapter IV was performed during actual flights in a Piper Cherokee and Douglas DC-3. These tests were done mainly to demonstrate the fact that the microcomputer system could be used in flight environments to provide navigation information. This flight testing also allowed the output of the coordinate converter to be recorded over a long period of time in flight situations for checking the stability of the microcomputer hardware and software system. Approximately 30 hours of flight testing were done and it was demonstrated that the output of the coordinate converter, in the form of range and bearing to a waypoint, could be useful to a pilot. Figure 5-3 shows a plot of the recorded data from the coordinate converter on a particular flight along with the recorded data from a commercial Loran-C receiver. From here it can be noted that there is an obvious bias in the microcomputer data on the east-west track to the north; this track coincides nearly with the LOP from the Seneca Carolina Beach pair, which was used for one of the TDs. The other pair used was Seneca-Dana, in the 99600 Loran-C chain. This particular flight path illustrates an interesting characteristic; the flight path along the north-south direction coincides more nearly to the Seneca-Dana LOP, which is a stronger signal pair, and the east-west track coincides with the weaker LOP signal pair. As is evident, the north-south track is on course and east-west track is slightly north of course. The commercial receiver flown incorporates signal path corrections, while the microcomputer system does not. It is obvious that signal path corrections are needed, although the inaccuracy without them amounts to one-half to one nautical mile.

The flight testing was done to assess the microcomputer-controlled navigation system's capabilities rather than evaluate absolute accuracy. Analysis of other flight tests similar to those shown in Figure 5-3 showed typical position uncertainties of one-half to one nautical mile. For general aviation use, this represents sufficient accuracy for enroute navigation and to place the pilot near navigation fixes at an airport for landing approaches. Much of the error noted

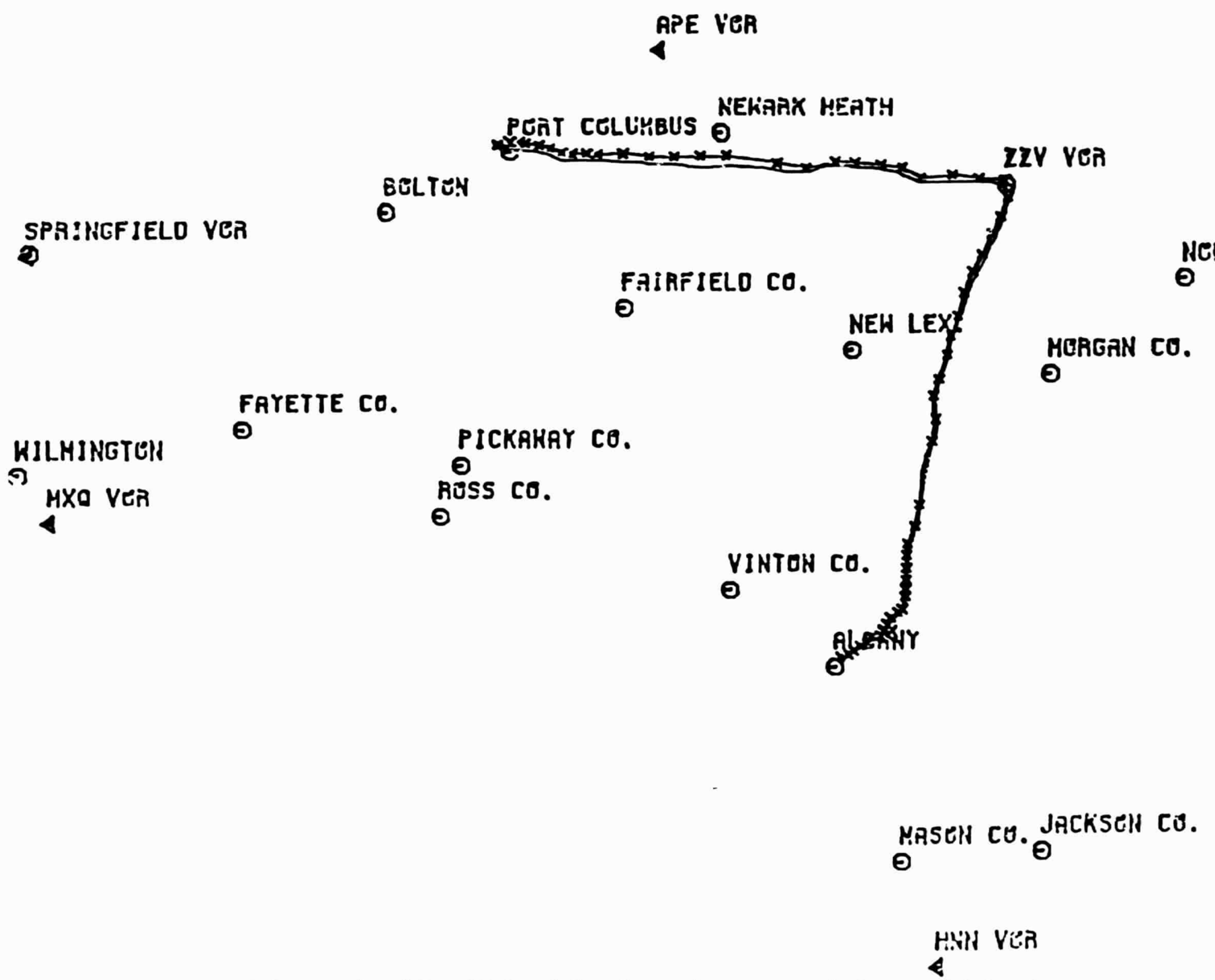

Figure 5-3. Results of Test Flight of Coordinate Converter with Commercial Receiver.
Scale: 1 inch = 7.5 nm.

in flight testing could be traced to the receiver where the TDs are derived. Assuming that the TDs could be derived correctly, the available accuracy would be on the order of 0.1 to 0.3 nm. The effects of the derivation of the TDs in the receiver is a function of how the pulse is sampled.

Flight testing also showed that the Loran-C signals may be received well at low altitudes, even in the hilly terrain where the tests were conducted. Effects on the ground such as 60 Hz power-line interference are much less pronounced during flight.

1. Errors as Function of Distance. Obviously, the choice of the central point used to compute the constants C1 through C9 in the spherical case, and C1 through C14 in the ellipsoidal case, has a definite bearing on how accurate the solution will be at any given point. If it is known that most operations will take place within a small part of a given triad coverage region, the central point could be chosen there. In general, though, the accuracy will deteriorate rapidly as the distance from the central point extends beyond 200 to 250 miles. The central point in the microcomputer program was chosen to correspond to the center of the triad region to provide good accuracy throughout the triad's coverage region.

B. Possible Fixes for Errors. Certainly, more accurate models of the earth profile over which the Loran-C signal travels would provide more accurate solutions. This becomes a very difficult task in practice because of all the various profiles which exist in any given area and it is not practical to store these and perform the necessary geodesic calculations on a small microcomputer system.

Many modern receiver designs are incorporating methods of applying propagation corrections to the received time differences as the position calculations are made. This is done by going to various locations and observing actual time differences. Propagation corrections become very important in determining how accurate any receiver system is, although as mentioned in Chapter II, these corrections typically are on the order of several tenths of a mile, thus the complexity of including propagation corrections may not be justified for some situations. It is also possible to include minor earth model corrections into the propagation corrections, although these would, in most cases, be very small corrections.

As was discussed above relating to Figure 5-3, the best way to provide more accurate TD-to-position calculations is to incorporate some sort of method of compensating for weak signals from the Loran-C stations. The strength of the signal is generally related to the signal path length and can thus provide useful information for other corrections, such as the profile of the earth over which the signal travelled. Although this can decrease inaccuracies from one nautical mile to less than one-tenth nautical mile, the added complexity of getting station signal-to-noise ratios from the Loran-C receiver and applying corrections based on these can sometimes be too much of a burden in processing time or storage requirements. It may be quite feasible to have several ground monitoring stations in the coverage region to determine local corrections (differential Loran) which could then be disseminated by some means to the pilot or operator and entered into the Loran-C navigation system.

ORIGINAL PAGE IS OF POOR QUALITY

VI. CONCLUSIONS AND RECOMMENDATIONS

In this paper, one objective has been to show the suitability of using Loran-C as a navigation system for general aviation use. Loran-C has many inherent capabilities which makes it attractive with respect to along and cross-track accuracy and signal coverage. Key conclusions reached as a result of this research are:

1. The obstacles to fully implementing a complete Loran-C navigator because of the hardware and the computational burdens are demonstrated to be removed with the wide availability of low-cost microprocessors and related devices.

2. The problem of converting Loran-C data to a more natural and easily understood form for the pilot using simple and fast techniques has been solved.

3. The cost of implementing a microprocessor-based Loran-C navigator unit can be comparable to the cost of a small microprocessor-based, general-purpose computer system in terms of hardware and software required.

4. The TD-to-position algorithm developed has comparable accuracy to that which can be achieved with raw Loran-C data; approximately 0.1 to 0.3 nm or less.

This work effort has concentrated on the computational aspects of a Loran-C navigator. The fact that transcendental functions are involved has impeded implementation of these ideas in the past; these functions are still difficult to implement on an eight-bit microprocessor. The tradeoffs between extra hardware to handle these functions or doing them in software must be made in terms of speed since other tasks must be done in the navigator. Thus, use of an external mathematics chip, such as the Am 9511, has a strong advantage in speed over software mathematics routines. This is especially true in light of recent price reductions for such satellite processors.

Because of velocities typical of aircraft operations, it is ideal to have complete position updates in one second. It should be noted that extra area-navigation computations in addition to the Loran-C position calculations (stated at 0.18 second) may exceed this desired value. The Am 9511 may be run asynchronously at speeds up to 2 MHz allowing the microprocessor time for other navigator tasks.

In view of the speed and accuracy of processor loading parameters of the coordinate conversion routine, the conclusion is that these techniques may be successfully used for general aviation Loran-C navigation application.

During the course of the investigation for this paper, several items were found which need to be addressed further. One is the acceptance of Loran-C as a primary means of navigation for aircraft. The present Loran-C system is maintained primarily for the maritime interests which means that some engineering changes may be necessary before full implementation of Loran-C for aviation could take place [37,38]. Additional stations are desired to cover fully the United States, and it is important to place these to provide maximum capability to users, especially those in areas of rough terrain or areas lacking in numbers of other services [39,40].

A very important issue is the consideration of how over-land propagation of the Loran-C signal affects the phase velocity of the signal. Explicit solutions to this are difficult to implement because of the very inconsistent nature of the propagation path. Empirical models are usually used. This is done by monitoring the Loran-C signal at many locations and averaging the results over long periods of time. A table of such results could be published and the values could be stored in the navigator unit, or broadcast to the aircraft much as barometric pressure readings are. Another important effect on the propagation path model is the geodetic model of the earth [41]. It is felt that these problems are manageable [42] and are not a serious impediment to fully using Loran-C as a primary means of air navigation.

VII. ACKNOWLEDGEMENTS

Acknowledgement is made to the following people who aided with the research reported in this paper: Dr. Robert W. Lilley, associate director, Mr. Ralph W. Burhans, project engineer, and student researchers Daryl McCall, Steven Yost and Stanley Novacki.

VIII. REFERENCES

[1]. Kayton, Myron and Walter R. Fried, "Avionics Navigation Systems," John Wiley and Sons, Inc., New York, NY, 1969, p.160.

[2] GPS Special Issue, Navigation, Vol. 25, No.2, Summer 1978.

[3]. Pursel, R. H., "Flight Test Investigation of LORAN-C for Civil Aviation Applications," Navigation, Vol. 27, No. 3, Fall 1980.

[4]. McConkey, E. D., "Evaluation of LORAN-C for Non-Precision Approach Applications," Navigation, Vol. 27, No. 3, Fall 1980.

[5]. Natarajan, Krishnan, "Testing of Loran-C For General Aviation Aircraft," NASA Conference Publication 2176, Proceedings of Joint University Program For Air Transportation Research - 1980 Conference, December 11-12, 1980.

[6]. Wong, Gene A., "Analysis of Loran-C System Reliability For Civil Aviation," Proceedings of Thirty-Seventh Annual Meeting, The Institute of Navigation, Annapolis, Maryland, June 1981.

[7]. Stremler, Ferrel G., "Introduction to Communication Systems," Addison-Wesley Publishing Company, Reading, Mass., 1979, p. 509.

[8]. Radio Amateur's Handbook, 51st edition, The American Radio Relay League, Newington, CT., 1974, pp. 559-66.

[9]. Swanson, E. R. and F. C. Robie, "VLF Navigation," NELC Technical Document 233, Naval Electronics Laboratory Center, San Diego, Cal., January 1973.

[10]. Lepanto, Janet A., "Omega Coordinate Transformations and Propagation Corrections," MS Thesis, Princeton University, Princeton, N. J., September 1980.

[11]. Chamberlin, Kent A., "Digital Correlation Detector for Low-Cost Omega Navigation," M.S. Thesis, Ohio University, Athens, Ohio, December 1975.

[12]. Pierce, J. A. and R. H. Woodward, "The Development of Long-Range Hyperbolic Navigation in the United States," Navigation, Vol. 18, No. 1, Spring 1971.

[13]. Hefley, Gifford, "The Development of Loran-C Navigation and Timing," NBS Monograph 129, U. S. Dept. of Commerce, National Bureau of Standards, Boulder, Colorado, October 1972.

[14]. Sherman, H. T. and V. L. Johnson, "The Loran-C Ground Station," Navigation, Vol. 23, No. 4, Winter 1976-77.

[15]. "Loran-C User Handbook," COMDINST MI6562.3, Department of Transportation, United States Coast Guard, May 1980.

[16]. "Coordinates and Emission Delays of Loran-C Stations," Time Service Announcement, Series 9, No. 150, U. S. Naval Observatory, Washington, DC, September 1979.

[17]. Op cit., Kayton and Fried, pp. 26-7.

[18]. "Method of Computing Loran Hyperbolic Lattice," Defense Mapping Agency, Hydrographic Center, Washington, D. C., 1977.

[19]. Holmstrom, J. S., "A New Approach to the Theory of Geodesics on an Ellipsoid," Navigation, Vol. 23, No. 3, Fall 1976.

[20]. Samaddar, S. N., "The Theory of Loran-C Ground Wave Propagation - A Review," Navigation, Vol. 26, No. 3, 1979.

[21]. Martin, O. L., Private Communication, Defense Mapping Agency, Washington, D. C., September, 1979.

[22]. Op cit., Kayton and Fried, pp. 27-8.

[23]. Campbell, A. C., "Loran to Geographic Conversion and Geographic to Loran Conversion," Informal Report N-3-64, Naval Oceanographic Office, Washington, D.C., June 1964.

[24]. Piecuch, L. M. and R. W. Lilley, "Interactive Loran-C to Geographic and Geographic-to-Loran-C Computation," NASA TM 52, Avionics Engineering Center, Ohio University, Athens, Ohio, August 1977.

[25]. Friedland, B. and M. F. Hutton, "New Algorithms For Converting Loran Time Differences to Position," Navigation, Vol. 20, No. 2, Summer 1973.

[26]. Roth, W. F., A. R. VanKoevering and D. Wozniak, "Hyperbolic Coordinate Converter," Lear Siegler Publication No. GRR-65-1023, Lear Siegler, Inc., Grand Rapids, Michigan, January 1965, pp. 3-1 to 3-6.

[27]. Razin, Sheldon, "Explicit (Noniterative) Loran Solution," Navigation, Vol. 14, No. 3, Fall 1967.

[28]. Fell, Harriet, "Comments on Loran Conversion Algorithms," Navigation, Vol. 22, No. 2, Summer 1975.

[29]. Op cit., Roth, W. F., et. al., pp. 3-7.

[30]. MCS6500 Microcomputer Family Hardware Manual, Publication Number 6500-50, MOS Technology, Inc., Norristown, PA, January 1976.

[31]. MCS6500 Microcomputer Family Programming Manual, Publication No. 6500-50, MOS Technology, Inc., Norristown, PA, August 1975.

[32]. Lilley, Robert W., "An Assembler For the MOS Technology 6502 Microprocessor as Implemented in Jolt(TM) and KIM-1(TM)," NASA TM 44, Avionics Engineering Center, Ohio University, Athens, Ohio, November 1976.

[33]. Lilley, R. W. and D. L. McCall, "A Loran-c Prototype Navigation Receiver For General Aviation," OU/NASA TM 80, Avionics Engineering Center, Ohio University, Athens, Ohio, August 1981.

[34]. Fischer, J. P. and R. W. Lilley, "A Video Display Interface For the Loran-C Navigation Receiver Development System," OU/NASA TM 58, Avionics Engineering Center, Athens, Ohio, May 1978.

[35]. "M6800 Microcomputer System Design Data," Motorola Semiconductor Products, Inc., Phoenix, Arizona, 1976.

[36]. Bucklen, Willard, John Eldon, Louis Schirm and Fred Williams, "Single-Chip Digital Multipliers Form Basic DSP Building Blocks," EDN, Vol. 26, No. 7, April 1, 1981.

[37]. Federal Radionavigation Plan, Volume 1, Radionavigation Plans and Policy and Volume 2, Requirements, Departments of Defense and Transportation, Washington, D. C., July, 1980.

[38]. Tyler, J. S., and F. G. Karkalik, "Radio Navigation Systems - Current Status," North Atlantic Treaty Organization Advisory Group For Aerospace Research and Development, AGARD-AG-245 (Principles and Operational Aspects of Precision Position Determination Systems), Neuilly-Sur-Seine, France, July 1979.

[39]. Polhemus, William L., "Operational and Economic Benefits Deriving From Use of Loran-C RNAV," Proceedings of the Thirty-Seventh Annual Meeting, The Institute of Navigation, Annapolis, Maryland, June 1981.

[40]. Rzonca, Lorraine, "Loran-C RNAV in Mountainous Areas," Proceedings of the Thirty-Seventh Annual Meeting, The Institute of Navigation, Annapolis, Maryland, June 1981.

[41]. Kuebler, W. and S. Sommers, "The Role of the Earth's Shape in Navigation: An Example," Navigation, Vol. 28, No. 1, Spring 1981.

[42] Walker, H.L. and R. Ellerbe, "Design Study Report for General Aviation Loran-C receiver," Teledyne Systems Company, U.S. Department of Transportation, Federal Aviation Administration, DOT FAA/RD-81/36, Washington, D.C., May 1981.

[43]. IBM System/360 and System/370 FORTRAN-IV Language, IBM Publication GC28-6515-10, May 1974.

IX. APPENDICES

Appendix I. Program listing for position-to-TD prediction program. This program is written in standard FORTRAN-IV computer language for the IBM/370-158 [43].

ORIGINAL PAGE IS
OF POOR QUALITY

```
C - THIS PROGRAM IS A TEST PROGRAM FOR COMPUTING LORAN-C
C - TIME DIFFERENCES.  THE EQUATION USED IS:
C -      TD = BETA + DELTA + TS - TM
C - WHERE BETA IS THE BASELINE LENGTH, AND DELTA IS THE
C - SECONDARY CODING DELAY.  TS AND TM MUST BE CALCULATED
C - AND REPRESENT THE BASELINE LENGTH FROM RECEIVER TO THE
C - SECONDARY STATION AND THE BASELINE LENGTH FROM THE
C - RECEIVER TO THE MASTER, RESPECTIVELY.  THESE TWO ARE
C - COMPUTED BY "ARC" AND THE ACCURACY DEPENDS ON THAT
C - SUBPROGRAM.  ALL GEODESIC CONSTANTS ARE NAD 27.
C -
C -                                     J.P.FISCHER   07/79
C -
C - CHAIN CONSTANTS ARE FOR U.S. EAST COAST (9960).
C -
      REAL*8 PHIR,GAMR,PHIM,GAMM,PHIW,GAMW,PHIX,GAMX,PHIY,GAMY,PHIZ,GAMZ
      COMMON PHIR,GAMR
      DATA PHIM,GAMM/0.7455002761,1.340870724/
      DATA PHIW,GAMW/0.8169491590,1.185559303/
      DATA PHIX,GAMX/0.7200063971,1.221345098/
      DATA PHIY,GAMY/0.5945057338,1.359840319/
      DATA PHIZ,GAMZ/0.6955476439,1.526928009/
      DATA CDBW/1.379724E4/,CDBX/2.696991E4/,CDBY/4.222161E4/,CDBZ/5.716
     >205E4/
C -
    5 PRINT 21
      READ(5,1,END=6)IDEG,MIN,SEC,IDEG1,MIN1,SEC1
      PHIR=(IDEG+(MIN+SEC/60.)/60.)*3.14159265/180.
      GAMR=(IDEG1+(MIN1+SEC1/60.)/60.)*3.14159265/180.
      TDM=ARC(PHIM,GAMM)
      TDW=CDBW+ARC(PHIW,GAMW)-TDM
      TDX=CDBX+ARC(PHIX,GAMX)-TDM
      TDY=CDBY+ARC(PHIY,GAMY)-TDM
      TDZ=CDBZ+ARC(PHIZ,GAMZ)-TDM
      PRINT 22,TDW,TDX,TDY,TDZ
      GOTO 5
    6 STOP
    1 FORMAT(I4,1X,I2,1X,F5.0)
   21 FORMAT('0ENTER RECEIVER LATITUDE/LONGITUDE.'/'  SDDD MM SS.SS')
   22 FORMAT(' W = ',F8.2,5X,'X = ',F8.2,5X,'Y = ',F8.2,5X,'Z = ',F8.2)
      END
      FUNCTION ARC(PHI1,XLNG2)
C
C     THIS PROGRAM IS DERIVED FROM KAYTON AND FRIED;
C     IT IS USED TO CALCULATE THE BASELINE DISTANCE BETWEEN TWO
C     POINTS ON THE REFERENCE ELLIPSOID BY DERIVING THE GEODESIC
C     ARC LENGTH IN A PLANE THROUGH THE CENTER OF THE ELLIPSOID.
C     REFERENCE ELLIPSOID USED IS CLARKE 1866 (NAD 27).
C
C     JP FISCHER 7/79
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*4 ARC
      COMMON PHI,XLNG1
      DATA A/6.378206406/,F/3.390075304D-3/
      DATA CL/299.792458/
C
      DXLNG=DABS(XLNG1-XLNG2)
      BETA=DATAN((1.-F)*DTAN(PHI))
      BETA1=DATAN((1.-F)*DTAN(PHI1))
      C1=DCOS(BETA1)*DSIN(DXLNG)
      C2=DCOS(BETA)*DSIN(BETA1)-DSIN(BETA)*DCOS(BETA1)*DCOS(DXLNG)
      C3=DSIN(BETA)*DSIN(BETA1)+DCOS(BETA)*DCOS(BETA1)*DCOS(DXLNG)
      PSI=DATAN(C1/C2)
      THETA=DATAN((C2*DCOS(PSI)+C1*DSIN(PSI))/C3)
      XM=(DSIN(BETA)+DSIN(BETA1))**2
      XN=((DSIN(BETA)-DSIN(BETA1))/DSIN(THETA))**2
      XU=(1.-DCOS(THETA))/DSIN(THETA)*(THETA-DSIN(THETA))/DSIN(THETA)
      XV=(1.+DCOS(THETA))*(THETA+DSIN(THETA))
      RHO=DABS(A*THETA-A*F*(XM*XU+XN*XV)/4.)
      ARC=RHO/CL
      RETURN
      END
```

Appendix II. Program listing for TD-to-lat/long conversion program. This program was written in standard FORTRAN-IV programming language and run on the IBM/370-158.

```
C     THIS PROGRAM USES AN EXPLICIT NON-ITERITIVE APPROACH TO SOLVE THE
C     LORAN SOLUTION OF CONVERTING TIME DIFFFRENCE READINGS TO LATITUDE
C     LONGITUDE.
C     EARTH CONSTANTS NAD-27 (CLARKE 1866 SPHERICAL EARTH MODEL)
C     11/1980 J.P. FISCHER
C     ELLIPSIOD CORRECTIONS APPLIED 6/1981
C
      DATA TCY,TCZ/3.9E-2,5.4E-2/
      DATA THMY/0.15129258/,THMZ/0.14849557/,XNR/46.986746/
      DATA CTMY/0.98857709/,STMY/0.15071607/,CTMZ/0.98899478/
      DATA STMZ/0.14795043/,CXK/0.37466368/,SXK/0.92716079/
      DATA C1/2.9832071/,C2/4.3111683/,C3/-7.1717116/
      DATA C4/4.0896360/,C5/-4.4643647/,C6/1.1496479/
      DATA C7/-5.0510869/,C8/5.1185063/,C9/.5592140O/
      DATA C10/1.5850077E-3/,C11/2.5836473E-4/,C12/-2.9670318E-3/
      DATA C13/1.5850077E-3/,C14/1.0034014/
C
    7 PRINT 12
      READ(5,3,END=10)TY,TZ
      TY=TY*1.E-6
      TZ=TZ*1.E-6
C
      PY=XNR*(TY-TCY)-THMY
      PZ=XNR*(TZ-TCZ)-THMZ
      CPY=COS(PY)
      SPY=SIN(PY)
      CPZ=COS(PZ)
      SPZ=SIN(PZ)
      AY=(CPY-CTMY)/STMY
      AZ=(CPZ-CTMZ)/STMZ
      BY=SPY/STMY
      BZ=SPZ/STMZ
      U1=AY*CXK-AZ
      U2=AY*SXK
      U3=AZ*BY-AY*BZ
      UU=U1*U1+U2*U2
      CDBY=(U3*U1+U2*SQRT(UU-U3*U3))/UU
      THMS=ATAN(AY/(BY+CDBY));
      CB=COS(THMS)
      CA=COS(THMS+PY)
      CC=COS(THMS+PZ)
C
      F=C1*CA+C2*CB+C3*CC
      G=C4*CA+C5*CB+C6*CC
      H=C7*CA+C8*CB+C9*CC
C
      THGS=ATAN((G+C10)/(F+C11))
      PHGS=ATAN(C14*C14*SIN(THGS)*(H+C12)/(G+C13))
      CALL LLRD(PHGS,THGS)
      GOTO 7
   10 STOP
    3 FORMAT(F10.0)
   12 FORMAT(' ENTER TIME-DIFFERENCES.')
      END
      SUBROUTINE RDLL(PHI,THE,*)
C
C     THIS SUBROUTINE CONVERTS GEOCENTRIC COORDINATES ENTERED BY THE
C     USER TO RADIAN COORDINATES.  INPUT FORM IS:  DDDD MM SS.SS
C     WHERE 'DDDD' IS THE DEGREES PORTION OF THE LAT. OR LONG., INCLUDIN
C     SIGN, MM IS THE MINUTES PORTION, AND SS.SS IS THE SECONDS PORTION.
C     READ FORMAT IS:  I4,1X,I2,1X,F5.0.
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*4 PHI,THE
      DATA PI/3.1415926535898/
      DATA MSG1/'LATI'/,MSG2/'TUDE'/,MSG3/':   '/,MSG4/'LONG'/,MSG5/'ITU
     >D'/,MSG6/'E:  '/
      PI1=PI/180.
C
C        PROMPT USER.
C
      PRINT1,MSG1,MSG2,MSG3
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      READ(5,10,END=21)ID1,IM1,SS1
      PHI=SNGL(PI1*(DFLOAT(ID1)+(DFLOAT(IM1)+SS1/60.)/60.))
C
C        PROMPT USER FOR LONGITUDE ENTRY.
C
      PRINT1,MSG4,MSG5,MSG6
      READ(5,10,END=21)ID1,IM1,SS1
      THE=SNGL(PI1*(DFLOAT(ID1)+(DFLOAT(IM1)+SS1/60.)/60.))
C
      RETURN
   21 RETURN 1
      ENTRY LLRD(PHI,THE)
C
C     THIS SUBSECTION DOES THE REVERSE OF THE ABOVE GIVEN RADIAN
C     COORDINATES.
C
      PI2=180./PI
      PHI=PI2*PHI
      THE=PI2*THE
      LD1=PHI
      XM1=(PHI-LD1)*60.
      LM1=XM1
      SS1=(XM1-LM1)*60.
C
      LD2=THE
      XM2=(THE-LD2)*60.
      LM2=XM2
      SS2=(XM2-LM2)*60.
      PRINT3,LD1,LM1,SS1,LD2,LM2,SS2
      RETURN
    1 FORMAT(' ENTER ',3A4/'  DDDD MM SS.SS')
   10 FORMAT(I4,1X,I2,1X,F5.0)
    3 FORMAT(' LATITUDE =  ',I4,1X,I2,1X,F5.2/' LONGITUDE = ',I4,1X,I2,1
     >X,F5.2)
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

Appendix III. Program listing for microprocessor version of TD-to-lat/long program. This program was written in standard MOS6502 assembly language and assembled by a cross assembler on the IBM/370-158.

```
*****************************************************************************
*                                                                           *
*      THIS PROGRAM IS DESIGNED TO CONVERT LORAN-C TIME-DIFFERENCES         *
*      TO LATITUDE/LONGITUDE AND RANGE/BEARING TO A WAYPOINT                *
*      ON THE MICROCOMPUTER USING THE AM9511A MATH CHIP.  THE               *
*      ALGORITHM IS BASED ON FORTRAN PROGRAM 'DEXLRN.'  BCD-TO-HEX          *
*      AND HEX-TO-BCD CONVERSIONS ARE MADE AT THE BEGINNING AND             *
*      END OF PROGRAM EXECUTION; ALL INTERNAL CALCULATIONS ARE              *
*      MADE USING BINARY FLOATING-POINT.  ALL SUBROUTINES ARE AT            *
*      THE END OF THE MAIN PROGRAM.  THE NUMBER TABLE AREA IS               *
*      DESIGNED TO BE PLACED AFTER THE SUBROUTINES; CONSTANTS ARE           *
*      FIRST, CALCULATED VARIABLES LAST.                                    *
*                                                                           *
*      VERSION 1.0, SPHERICAL MODEL, 11/1980, J.P.FISCHER                   *
*      VERSION 2.0, ELLIPSOID MODEL, 7/1981, J. P. FISCHER                  *
*                                                                           *
*****************************************************************************
*
*
PIAA    EQU $9000       PERIPHERAL AND DDR SIDE A
PIAB    EQU $9002       PERIPHERAL AND DDR SIDE B
*
*          AM9511A COMMANDS.
*
FADD    EQU $10
FSUB    EQU $11
FMUL    EQU $12
FDIV    EQU $13
SQRT    EQU 1
SIN     EQU 2
COS     EQU 3
ATAN    EQU 7
PTOF    EQU $17
PUPI    EQU $1A
FLTD    EQU $1C
FIXD    EQU $1E
*
*           VARIABLE NAME TABLE
*
*
*           THE FOLLOWING ARE CONSTANTS.
*
TCY     EQU 0
TCZ     EQU TCY+4
THMY    EQU TCZ+4
THMZ    EQU THMY+4
XNR     EQU THMZ+4
CTMY    EQU XNR+4
STMY    EQU CTMY+4
CTMZ    EQU STMY+4
STMZ    EQU CTMZ+4
CXK     EQU STMZ+4
SXK     EQU CXK+4
C1      EQU SXK+4
C2      EQU C1+4
C3      EQU C2+4
C4      EQU C3+4
C5      EQU C4+4
C6      EQU C5+4
C7      EQU C6+4
C8      EQU C7+4
C9      EQU C8+4
C10     EQU C9+4
C11     EQU C10+4
C12     EQU C11+4
C14     EQU C12+4
EM6     EQU C14+4
RCR     EQU EM6+4
TWO     EQU RCR+4
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C256    EQU TWO+4
P180    EQU C256+4
C60     EQU P180+4
PHI     EQU C60+4
RLAM    EQU PHI+4
*
*       THE FOLLOWING ARE CALCULATED VARIABLES.
*
TY      EQU RLAM+4
TZ      EQU TY+4
PY      EQU TZ+4
PZ      EQU PY+4
CPY     EQU PZ+4
SPY     EQU CPY+4
CPZ     EQU SPY+4
SPZ     EQU CPZ+4
AY      EQU SPZ+4
AZ      EQU AY+4
BY      EQU AZ+4
BZ      EQU BY+4
U1      EQU BZ+4
U2      EQU U1+4
U3      EQU U2+4
UU      EQU U3+4
CDBY    EQU UU+4
THMS    EQU CDBY+4
CB      EQU THMS+4
CA      EQU CB+4
CC      EQU CA+4
F       EQU CC+4
G       EQU F+4
H       EQU G+4
THGS    EQU H+4
PHGS    EQU THGS+4
CF1     EQU PHGS+4
DLAM    EQU CF1+4
DPHI    EQU DLAM+4
TEMP    EQU DPHI+4
*
*       PAGE-ZERO ASSIGNMENTS
*
        ORG 0
TDA     BSS 4          PACKED BCD /W TENTH DIGIT
        BSS 4          PACKED BCD /W TENTH DIGIT
        ORG $BE
AGCF    BSS 1
AGCB    BSS 1
BASE1   BSS 2          BASE ADDRESS FOR NUMBER MOVE
XLIM    BSS 1          INDEX LIMIT USED IN NUMBER CONVERSION ROUTINES
XTEMP   BSS 1          SAVE AREA FOR X-REGISTER
YTEMP   BSS 1          SAVE AREA FOR Y-REGISTER
COUNT   BSS 1          COUNT-DOWN REGISTER
VY      BSS 1
FLAG    BSS 1
HEX     BSS 3          TABLE FOR BUILDING UP HEX NUMBER FROM BCD
RES     BSS 3          RESIDUE TABLE FOR BINARY MULTIPLICATION
DVDN    BSS 2          HEX NUMBER TO BE CONVERTED TO BCD
DRES    BSS 5          RESULT OF HEX-TO-BCD CONVERSION
DVSR    BSS 1          CONSTANT USED IN BINARY DIVISION
RMNDR   BSS 1          REMAINDER FROM BINARY DIVISION
*
*       THE FOLLOWING ARE THE
*       COMPUTED LATITUDE/LONGITUDE AND RANGE/BEARING.  ALL
*       ARE IN PACKED BCD FORMAT.  THESE ARE DESIGNED TO
*       INTERFACE TO SENSOR SOFTWARE.
*
LAT     BSS 3          FORMAT: (EX.) 39 19 20 - DEGREES, MINUTES, SECON
        BSS 3          FORMAT: (EX.) 82 05 56 - DEGREES, MINUTES, SECON
RHO     BSS 2          FORMAT: (EX.) 00 96 - NAUTICAL MILES /W TENTH DI
```

```
        BSS 2           FORMAT: (EX.) 22 04 - DEGREES /W TENTH DIGIT
BASE    BSS 2           ADDRESS OF FLOATING-POINT TABLE
*
        ORG $1700
*
*           CONVERT TD'S FROM BCD TO FLOATING-POINT
*
        LDX =0
        LDY =TY         CONVERT TDA FIRST
        STY YTEMP
        LDA =2
        STA COUNT       CONVERT TWO TDS
        CLD             SET DECIMAL MODE OFF
*
*           BCD-TO-HEX CONVERSION
*
TOHEX   LDY =6          SIX DIGITS
        LDA =0
        STA HEX
        STA HEX+1
        STA HEX+2
        STA FLAG
        JMP LFOUR       DO LOWER PART FIRST
UFOUR   DEC FLAG
        LDA TDA,X       GET A BYTE
        LSR A
        LSR A
        LSR A
        LSR A
        JMP TO3         DO THE MAIN CONVERSION
LFOUR   INC FLAG
        LDA TDA,X       GET A BYTE
        AND =$F         REMOVE UPPER FOUR BITS
        INX
TO3     CLC             FOR ADDITION
        ADC HEX+2
        STA HEX+2       ADD DIGIT TO PARTIAL SUM
        BCC NOC         GO IF NO CARRY OUT
        INC HEX+1
NOC     DEY             NEXT DIGIT
        BEQ TOD         IF DONE, LEAVE
        STY XTEMP       SAVE COUNTER
*
*           MULTIPLY PARTIAL SUM BY TEN
*
        LDA =0
        STA RES         CLEAR MULT. TABLE
        STA RES+1
        STA RES+2
        LDA =10         DIVISOR
        LDY =8
TO2     CLC
        ROL RES+2
        ROL RES+1
        ROL RES
        ASL A
        BCC NOC2
        PHA
        CLC
        LDA RES+2
        ADC HEX+2
        STA RES+2
        LDA RES+1
        ADC HEX+1
        STA RES+1
        LDA RES
        ADC HEX
        STA RES
        PLA
```

```
NOC2    DEY
        BNE TO2
        LDY XTEMP
        LDA RES
        STA HEX
        LDA RES+1
        STA HEX+1
        LDA RES+2
        STA HEX+2
        LDA FLAG
        BNE UFOUR
        JMP LFOUR
*
*           NOW CHANGE INTEGER PART TO FLOATING-POINT
*           THEN ADD IN THE FRACTIONAL PART.
*
TOD     LDY YTEMP
        STX XTEMP         SAVE THE CURRENT DIGIT LOCATION
        LDA =0
        STA (BASE),Y      CLEAR THE UPPER TWO BYTES
        INY
        LDA HEX           MSB OF HEX INTEGER
        STA (BASE),Y      PUT IN TABLE FOR 9511
        INY
        LDA HEX+1         LSB OF HEX INTEGER
        STA (BASE),Y
        INY
        LDA HEX+2
        STA (BASE),Y
        LDY YTEMP         POINT TO TD NUMBER
        JSR PUSH          GIVE IT TO 9511
        LDA =FLTD
        JSR CMND          CONVERT INTEGER TO FLOATING-POINT
        LDY =EM6          CONSTANT 1E-7
        JSR PUSH
        LDA =FMUL         CONVERT FROM MICROSECONDS...
        JSR CMND          TO SECONDS
        LDY YTEMP         GET THE TD LOCATION AGAIN
        JSR POP           AND STORE THE TD
        DEC COUNT         SEE IF BOTH TDS CONVERTED
        BEQ SUTD          IF SO, START TD-TO-POSITION CONVERSION
        LDX XTEMP         SET UP FOR NEXT TD
        INX
        LDY =TZ           ADDRESS OF TDB
        STY YTEMP         STORE IT
        JMP TOHEX         REPEAT
*
*           CALCULATE 'PY'
*
SUTD    LDY =XNR          'XNR'
        JSR PUSH          PUSH ON STACK
        LDY =TY           'TY'
        JSR PUSH
        LDY =TCY          'TCY'
        JSR PUSH
        LDA =FSUB
        JSR CMND          SUBTRACT TY-TYC
        LDA =FMUL
        JSR CMND          XNR*(TY-TYC)
        LDY =THMY         'THMY'
        JSR PUSH
        LDA =FSUB
        JSR CMND          XNR*(TY-TZ)-THMY
        LDY =PY           'PY'
        JSR POP           PUT 'PY' INTO TABLE
*
*           CALCULATE 'PZ'
*
        LDY =XNR          'XNR'
```

```
         JSR PUSH       PUSH ONTO STACK
         LDY =TZ        'TZ'
         JSR PUSH       PUSH ONTO STACK
         LDY =TCZ       'TCZ'
         JSR PUSH       PUSH ONTO STACK
         LDA =FSUB
         JSR CMND       TZ-TCZ
         LDA =FMUL
         JSR CMND       XNR*(TZ-TCZ)
         LDY =THMZ      'THMZ'
         JSR PUSH
         LDA =FSUB
         JSR CMND       XNR*(TZ-TCZ)-THMZ
         LDY =PZ        LOCATION FOR 'PZ'
         JSR POP        STORE RESULT IN 'PZ'
*
*        CALCULATE CPY,SPY,CPZ,SPZ
*
         LDY =PY        'PY'
         JSR PUSH       PUSH IT
         LDY =PY        'PY'
         JSR PUSH       DUPLICATE STACK
         LDA =COS
         JSR CMND       COS(PY)
         LDY =CPY       FOR CPY
         JSR POP        GET IT
         LDA =SIN
         JSR CMND       SIN(PY)
         LDY =SPY
         JSR POP        GET SPY
         LDY =PZ        'PZ'
         JSR PUSH
         LDY =PZ        'PZ'
         JSR PUSH
         LDA =COS
         JSR CMND       COS(PZ)
         LDY =CPZ       LOCATION FOR CPZ
         JSR POP
         LDA =SIN
         JSR CMND       SIN(PZ)
         LDY =SPZ       LOCATION FOR 'SPZ'
         JSR POP
*
*        CACULATE 'AY'
*
         LDY =CPY       'CPY'
         JSR PUSH
         LDY =CTMY      'CTMY'
         JSR PUSH
         LDA =FSUB
         JSR CMND       CPY-CTMY
         LDY =STMY      'STMY'
         JSR PUSH
         LDA =FDIV
         JSR CMND       (CPY-CTMY)/STMY
         LDY =AY        'AY'
         JSR POP
*
*        CALCULATE 'AZ'
*
         LDY =CPZ       'CPZ'
         JSR PUSH
         LDY =CTMZ      'CTMZ'
         JSR PUSH
         LDA =FSUB
         JSR CMND       CPZ-CTMZ
         LDY =STMZ      'STMZ'
         JSR PUSH
         LDA =FDIV
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
         JSR CMND          (CPZ-CTMZ)/STMZ
         LDY =AZ           LOCATION FOR 'AZ'
         JSR POP
*
*           CALCULATE 'BY'
*
         LDY =SPY          'SPY'
         JSR PUSH
         LDY =STMY         'STMY'
         JSR PUSH
         LDA =FDIV
         JSR CMND          SPY/STMY
         LDY =BY           LOCATION FOR 'BY'
         JSR POP           GET IT
*
*           CALCULATE 'BZ'
*
         LDY =SPZ          'SPZ'
         JSR PUSH
         LDY =STMZ         'STMZ'
         JSR PUSH
         LDA =FDIV
         JSR CMND          SPZ/STMZ
         LDY =BZ           LOCATION FOR 'BZ'
         JSR POP           GET IT
*
*           CALCULATE 'U1'
*
         LDY =AY           'AY'
         JSR PUSH
         LDY =CXK          'CXK'
         JSR PUSH
         LDA =FMUL
         JSR CMND          AY*CXK
         LDY =AZ           'AZ'
         JSR PUSH
         LDA =FSUB
         JSR CMND          AY*CXK-AZ
         LDY =U1           LOCATION FOR 'U1'
         JSR POP
*
*           CLACULATE 'U2'
*
         LDY =AY           'AY'
         JSR PUSH
         LDY =SXK          'SXK'
         JSR PUSH
         LDA =FMUL
         JSR CMND          AY*SXK
         LDY =U2
         JSR POP           GET 'U2'
*
*           CALCULATE 'U3'
*
         LDY =AZ           'AZ'
         JSR PUSH
         LDY =BY           'BY'
         JSR PUSH
         LDA =FMUL
         JSR CMND          AZ*BY
         LDY =AY           'AY'
         JSR PUSH
         LDY =BZ           'BZ'
         JSR PUSH
         LDA =FMUL
         JSR CMND          AY*BZ
         LDA =FSUB
         JSR CMND          AZ*BY-AY*BZ
         LDY =U3           LOCATION FOR 'U3'
```

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

```
      JSR POP
*
*         CALCULATE 'UU'
*
      LDY =U1         'U1'
      JSR PUSH
      LDY =U1         'U1'
      JSR PUSH
      LDA =FMUL
      JSR CMND        U1*U1
      LDY =U2         'U2'
      JSR PUSH
      LDY =U2
      JSR PUSH
      LDA =FMUL
      JSR CMND        U2*U2
      LDA =FADD
      JSR CMND        U1*U1+U2*U2
      LDY =UU         LOCATION FOR 'UU'
      JSR POP
*
*         CALCULATE 'CDBY'
*
      LDY =UU         'UU'
      JSR PUSH
      LDY =U3         'U3'
      JSR PUSH
      LDY =U3         'U3'
      JSR PUSH
      LDA =FMUL
      JSR CMND        U3*U3
      LDA =FSUB
      JSR CMND        UU-U3*U3
      LDA =SQRT
      JSR CMND        SQRT(UU-U3*U3)
      LDY =U2         'U2'
      JSR PUSH
      LDA =FMUL
      JSR CMND        U2*SQRT(UU-U3*U3)
      LDY =U3         'U3'
      JSR PUSH
      LDY =U1         'U1'
      JSR PUSH
      LDA =FMUL
      JSR CMND        U3*U1
      LDA =FADD
      JSR CMND        U3*U1+U2*SQRT(UU-U3*U3)
      LDY =UU         'UU'
      JSR PUSH
      LDA =FDIV
      JSR CMND        (U3*U1+U2*SQRT(UU-U3*U3))/UU
      LDY =CDBY       LOCATION FOR 'CDBY'
      JSR POP
*
*         CALCULATE 'THMS'
*
      LDY =AY         'AY'
      JSR PUSH
      LDY =BY         'BY'
      JSR PUSH
      LDY =CDBY       'CDBY'
      JSR PUSH
      LDA =FADD
      JSR CMND        BY+CDBY
      LDA =FDIV
      JSR CMND        AY/(BY+CDBY)
      LDA =ATAN
      JSR CMND        ATAN(AY/(BY+CDBY))
      LDY =THMS       LOCATION FOR 'THMS'
```

```
         JSR POP
*
*            CALCULATE 'CB'
*
         LDY =THMS         'THMS'
         JSR PUSH
         LDA =COS
         JSR CMND          COS(THMS)
         LDY =CB           LOCATION FOR 'CB'
         JSR POP
*
*            CALCULATE 'CA'
*
         LDY =THMS         'THMS'
         JSR PUSH
         LDY =PY           'PY'
         JSR PUSH
         LDA =FADD
         JSR CMND          THMS+PY
         LDA =COS
         JSR CMND          COS(THMS+PY)
         LDY =CA           LOCATION FOR 'CA'
         JSR POP
*
*            CALCULATE 'CC'
*
         LDY =THMS         'THMS'
         JSR PUSH
         LDY =PZ           'PZ'
         JSR PUSH
         LDA =FADD
         JSR CMND          THMS+PZ
         LDA =COS
         JSR CMND          COS(THMS+PZ)
         LDY =CC           LOCATION FOR 'CC'
         JSR POP
*
*            CALCULATE 'F'
*
         LDY =C1           'C1'
         JSR PUSH
         LDY =CA           'CA'
         JSR PUSH
         LDA =FMUL
         JSR CMND          C1*CA
         LDY =C2           'C2'
         JSR PUSH
         LDY =CB           'CB'
         JSR PUSH
         LDA =FMUL
         JSR CMND          C2*CB
         LDA =FADD
         JSR CMND          C1*CA+C2*CB
         LDY =C3           'C3'
         JSR PUSH
         LDY =CC           'CC'
         JSR PUSH
         LDA =FMUL
         JSR CMND          C3*CC
         LDA =FADD
         JSR CMND          C1*CA+C2*CB+C3*CC
         LDY =F            LOCATION FOR 'F'
         JSR POP
*
*            CALCULATE 'G'
*
         LDY =C4           'C4'
         JSR PUSH
         LDY =CA           'CA'
```

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

```
         JSR PUSH
         LDA =FMUL
         JSR CMND          C4*CA
         LDY =C5           'C5'
         JSR PUSH
         LDY =CB           'CB'
         JSR PUSH
         LDA =FMUL
         JSR CMND          C5*CB
         LDA =FADD
         JSR CMND          C4*CA+C5*CB
         LDY =C6           'C6'
         JSR PUSH
         LDY =CC           'CC'
         JSR PUSH
         LDA =FMUL
         JSR CMND          C6*CC
         LDA =FADD
         JSR CMND          C4*CA+C5*CB+C6*CC
         LDY =G            'G'
         JSR POP           GET 'G'
*
*            CALCULATE 'H'
*
         LDY =C7           'C7'
         JSR PUSH
         LDY =CA           'CA'
         JSR PUSH
         LDA =FMUL
         JSR CMND          C7*CA
         LDY =C8           'C8'
         JSR PUSH
         LDY =CB           'CB'
         JSR PUSH
         LDA =FMUL
         JSR CMND          C8*CB
         LDA =FADD
         JSR CMND          C7*CA+C8*CB
         LDY =C9           'C9'
         JSR PUSH
         LDY =CC           'CC'
         JSR PUSH
         LDA =FMUL
         JSR CMND          C9*CC
         LDA =FADD
         JSR CMND          C7*CA+C8*CB+C9*CC
         LDY =H            LOCATION 'H'
         JSR POP
*
*            CALCULATE 'THGS'
*
         LDY =G            'G'
         JSR PUSH
         LDY =C10          'C10'
         JSR PUSH
         LDA =FADD
         JSR CMND          G+C10
         LDY =F            'F'
         JSR PUSH
         LDY =C11          'C11'
         JSR PUSH
         LDA =FADD
         JSR CMND          F+C11
         LDA =FDIV
         JSR CMND          (G+C10)/(F+C11)
         LDA =ATAN
         JSR CMND          ATAN((G+C10)/(F+C11))
         LDA =PTOF
         JSR CMND          DUPLICATE STACK
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        LDY =THGS
        JSR POP           GET THGS
        LDY =P180         180/PI
        JSR PUSH
        LDA =FMUL
        JSR CMND          CONVERT FROM RADIANS TO DEGREES
        LDX =3            POINT TO LONGITUDE FIELD
        LDA =6            INDEX LIMIT
        STA XLIM
        LDA =$8D          VIDEO LOCATION FOR LONG.
        STA VY
        JSR TOBCD2        CONVERT TO DEGREES, MINUTES, SECONDS
*
*           CALCULATE 'PHGS'
*
        LDY =THGS         'THGS'
        JSR PUSH
        LDA =SIN
        JSR CMND          SIN(THGS)
        LDY =C14
        JSR PUSH
        LDY =C14
        JSR PUSH
        LDA =FMUL
        JSR CMND          C14*C14
        LDA =FMUL
        JSR CMND          C14*C14*SIN(THGS)
        LDY =H            'H'
        JSR PUSH
        LDY =C12          'C12'
        JSR PUSH
        LDA =FADD
        JSR CMND          H+C12
        LDA =FMUL
        JSR CMND          C14*C14*SIN(THGS)*(H+12)
        LDY =G            'G'
        JSR PUSH
        LDY =C10          'C10'
        JSR PUSH
        LDA =FADD
        JSR CMND          G+C13
        LDA =FDIV
        JSR CMND          C14*C14*SIN(THGS)*(H+C12)/(G+C10)
        LDA =ATAN
        JSR CMND          ARCTAN(C14*C14*SIN(THGS)*(H+C12)/(G+C13))
        LDA =PTOF
        JSR CMND          DUPLICATE STACK LOCATIONS
        LDY =PHGS
        JSR POP           GET PHGS
        LDY =P180         180/PI
        JSR PUSH
        LDA =FMUL
        JSR CMND          CONVERT PHGS FROM RADIANS TO DEGREES
        LDX =0            LOCATION FOR LATITUDE FIELD
        LDA =3            INDEX LIMIT
        STA XLIM
        LDA =$6D          VIDEO LOCATION FOR LAT.
        STA VY
        JSR TOBCD2        CONVERT TO DEGREES, MINUTES, SECONDS
*
*           RHO-THETA CONVERSION
*           CALCULATE 'CFI'
*
        LDY =PHGS         'PHGS'
        JSR PUSH
        LDY =PHI          'PHI'
        JSR PUSH
        LDA =FADD
        JSR CMND          PHGS+PHI
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        LDY =TWO        "2"
        JSR PUSH
        LDA =FDIV
        JSR CMND        (PHGS+PHI)/2
        LDA =COS
        JSR CMND        COS((PHGS+PHI)/2)
        LDY =CF1        LOCATION FOR 'CF1'
        JSR POP
*
*       CALCULATE 'DLAM'
*
        LDY =THGS       'THGS'
        JSR PUSH
        LDY =RLAM       'RLAM'
        JSR PUSH
        LDA =FSUB
        JSR CMND        THGS-RLAM
        LDY =DLAM       LOCATION FOR 'DLAM'
        JSR POP         GET IT
*
*       CALCULATE 'DPHI'
*
        LDY =PHI        'PHI'
        JSR PUSH
        LDY =PHGS       'PHGS'
        JSR PUSH
        LDA =FSUB
        JSR CMND        PHI-PHGS
        LDY =DPHI       LOCATION FOR 'DPHI'
        JSR POP
*
*       CALCULATE 'ARC'
*
        LDY =DLAM       'DLAM'
        JSR PUSH
        LDY =CF1        'CF1'
        JSR PUSH
        LDA =FMUL
        JSR CMND        DLAM*CF1
        LDA =PTOF
        JSR CMND        DUPLICATE STACK LOCATIONS
        LDA =FMUL
        JSR CMND        (DLAM*CF1)**2
        LDY =DPHI       'DPHI'
        JSR PUSH
        LDY =DPHI
        JSR PUSH        'DPHI'
        LDA =FMUL
        JSR CMND        DPHI*DPHI
        LDA =FADD
        JSR CMND        (DLAM*CF1)**2+DPHI*DPHI
        LDA =SQRT
        JSR CMND        SQRT( ... )
        LDY =RCR        'RCR'
        JSR PUSH
        LDA =FMUL
        JSR CMND        RCR*SQRT( ... )
        LDX =0          POINT TO RANGE FIELD
        LDA =$B0        VIDEO LOCATION FOR RANGE
        STA VY
        JSR RNGB        GET RANGE IN BCD
*
*       CALCULATE 'PSI'
*
        LDY =DLAM       'DLAM'
        JSR PUSH
        LDY =CF1        'CF1'
        JSR PUSH
        LDA =FMUL
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        JSR CMND         DLAM*CF1
        LDY =DPHI        'DPHI'
        JSR PUSH
        LDA =FDIV
        JSR CMND         DLAM*CF1/DPHI
        LDA =ATAN
        JSR CMND         ATAN(DLAM*CF1/DPHI)
*
*           ADJUST 'PSI' FOR PROPER QUADRANT
*
        LDY =DPHI
        LDA (BASE),Y     GET THE SIGN OF 'DPHI'
        BMI L30          GO IF NEGATIVE
        LDY =DLAM
        LDA (BASE),Y     GET SIGN OF 'DLAM'
        BPL L33          GO IF POSITIVE
*
        LDA =PUPI
        JSR CMND         PUSH PI
        LDA =PUPI
        JSR CMND         PUSH PI
        LDA =FADD
        JSR CMND         MULTIPLY PI BY TWO
        LDA =FADD
        JSR CMND         PSI + 2*PI
        JMP L33
*
L30     LDA =PUPI
        JSR CMND         PUSH PI
        LDA =FADD
        JSR CMND         PSI+PI
*
L33     LDY =P180        180/PI
        JSR PUSH
        LDA =FMUL
        JSR CMND         CONVERT TO DEGREES
        LDX =2           LOCATION FOR BEARING
        LDA =$D0         VIDEO LOCATION FOR BEARING
        STA VY
        JSR RNGB
*
*           DONE
*
        SED              SET DECIMAL MODE FOR SENSOR
        RTS
*
*
************************************************************************
*                                                                      *
*           THIS SUBROUTINE INITIALIZES THE PIA FOR USE WITH           *
*           THE MATH CHIP AND THEN SETS THE CONTROL INPUTS OF THE      *
*           MATH CHIP TO INACTIVE STATES.                              *
*                                                                      *
************************************************************************
*
PINT    LDA PIAA         CLEAR INTERRUPTS
        LDA PIAB
        LDA =$14
        STA PIAA+1       SET INTERRUPT CONTROL AND DDR
        LDA =0
        STA PIAB+1       SET DDR LOW
        LDA =$CF
        STA PIAB         SET INPUTS AND OUTPUTS FOR 9511
        LDA =4
        STA PIAB+1       SET DDR BIT HIGH
        LDA =7
        STA AGCB         SET BACKGROUND COORDINATION BYTE
        STA PIAB         SET CD, RD AND WR HIGH
        LDA =$F
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        STA PIAB        SET SVACK HIGH
        LDA PIAA        CLEAR ANY INTERRUPTS
        LDA =0          GET LSB OF NUMBER TABLE
        STA BASE        SAVE FOR INDEXING
        LDA =2          GET MSB
        STA BASE+1
        LDA =$A
        STA DVSR        STORE 10 FOR BINARY DIVISION
*
*           MOVE NUMBER TABLE TO READ/WRITE SPACE
*
        LDY =0          CLEAR INDEX ONE
        LDX =0          CLEAR INDEX TWO
        LDA TABLE
        STA BASE1
        LDA TABLE+1
        STA BASE1+1
MVE1    LDA (BASE1),Y   GET A DIGIT
        STA $200,X      STORE IT IN NEW LOCATION
        INX
        INY
        CPY =120        MOVE 120 YET?
        BMI MVE1        IF NOT, CONTINUE
*
*           INITIALIZE VIDEO DISPLAY OUTPUT
*
        LDX =0
WRSCN   LDY SCLC,X
        LDA LSCRN,X
        AND =$3F
        STA $2200,Y
        INX
        CPX =16
        BMI WRSCN
*
        RTS
*
*
***********************************************************************
*                                                                     *
*           SUBROUTINE TO SEND DATA TO 9511.                          *
*           ITEM NUMBER IS IN REG-Y.  BASE ADDRESS                    *
*           IS IN 'BASE.'                                             *
*                                                                     *
***********************************************************************
*
PUSH    JSR SAO         SET PIAA TO OUTPUTS
        INY             ADJUST REG-Y SO IT
        INY             POINTS TO LSB OF NUMBER
        INY
        LDX =4          LOAD COUNT OF 4
PSH1    LDA (BASE),Y    GET A BYTE OF THE NUMBER
        STA PIAA        GIVE IT TO 9511
        LDA =$A
        JSR SV10
        LDA PIAA        CLEAR ANY INTERRUPTS
        DEY             NEXT BYTE
        DEX
        BNE PSH1        LOOP UNTIL ENTIRE WORD WRITTEN
        RTS             IF DONE, RETURN
*
*
***********************************************************************
*                                                                     *
*           SUBROUTINE SETS UP PIA SIDE A AS OUTPUTS.                 *
*                                                                     *
***********************************************************************
*
SAO     LDA PIAA+1      GET THE CONTROL REGESTER
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        AND =$FB        SET ACCESS THE DDR
        STA PIAA+1      RETURN IT
        LDA =$FF
        STA PIAA        SET ALL TO OUTPUTS
        LDA PIAA+1
        ORA =4          SET DDR BIT HIGH
        STA PIAA+1      RETURN IT
        RTS
*
*******************************************************************
*                                                                 *
*       THIS SUBROUTINE POPS A NUMBER OFF OF THE                  *
*       9511 STACK.  NUMBER IS RETURNED TO LOCATION               *
*       WITH 'BASE' AS BASE ADDRESS; ITEM NUMBER IN Y.            *
*                                                                 *
*******************************************************************
*
POP     JSR SAI         SET PIA AS INPUTS
        LDX =4          LOAD COUNT OF 4
POP1    LDA =9
        JSR SV9
        STA (BASE),Y    STORE IN TABLE
        LDA =$B
        STA AGCB
        ORA AGCF
        STA PIAB        SET RD HIGH TO INCR. STACK POINTER
        INY
        DEX
        BNE POP1        DO 4 BYTES
        RTS
*
*
*******************************************************************
*                                                                 *
*         SUBROUTINE SETS UP PIAA AS INPUTS.                      *
*                                                                 *
*******************************************************************
*
SAI     LDA PIAA+1
        AND =$FB        SET DDR BIT LOW
        STA PIAA+1
        LDA =0
        STA PIAA        SET SIDE A TO ALL INPUTS
        LDA PIAA+1
        ORA =4          SET DDR BIT HIGH
        STA PIAA+1
        RTS
*
*
*******************************************************************
*                                                                 *
*         SUBROUTINE SENDS THE COMMAND BYTE IN ACCUM.             *
*         TO THE 9511 FOR EXECUTION.  ROUTINE RETURNS TO          *
*         CALLER REGARDLESS OF WHETHER EXECUTION IS COMPLETED     *
*         OR NOT.                                                 *
*                                                                 *
*******************************************************************
*
CMND    PHA
        JSR SAO         SET PIA SIDE A AS OUTPUTS
        PLA             GET THE COMMAND
        STA PIAA        SEND TO 9511
        LDA =$E
        JSR SV10
        BIT PIAA+1      TEST IF COMMAND DONE
        BPL *-3         KEEP TESTING UNTIL DONE
        LDA PIAA        CLEAR THE INTERRUPT BIT
*
*         READ THE STATUS REGISTER; RETURN IF INVALID CODE
```

```
*            WAS PRODUCED.
*
        JSR SAI             SET PIAA SIDE A AS INPUTS
        LDA =$D             SET C/D HIGH, RD LOW
        JSR SV9
        PHA                 SAVE IT
        LDA =$F             RETURN 9511 TO INACTIVE STATE
        STA AGCB
        ORA AGCF
        STA PIAB
        PLA
        AND =%10011110      ZERO OUT UNIMPORTANT BITS
        BEQ OK              IF ZERO, CONTINUE PROCESSING
        PLA                 POP THE STACK
        PLA
        SED                 SET DECIMAL FOR SENSOR ROUTINE
OK      RTS                 RETURN
*
*
SV9     STA AGCB
        ORA AGCF
        STA PIAB
        BIT PIAA+1
        BVC *-3
        LDA PIAA
        RTS
*
*
SV10    PHA
        STA AGCB
        ORA AGCF
        STA PIAB
        PLA
        CLC
        ADC =1
        STA AGCB
        ORA AGCF
        STA PIAB
        RTS
*
*
*******************************************************************************
*                                                                             *
*       THIS ROUTINE CONVERTS THE BINARY FLOATING-POINT LATITUDE              *
*       AND LONGITUDE SEPARATELY INTO THE STANDARD DEGREE, MINUTES,           *
*       AND SECOND FORMAT.  THE RESULT IS STORED IN BCD.                      *
*                                                                             *
*******************************************************************************
*
INTG    CLD                 SET DECIMAL MODE OFF
        LDA =PTOF
        JSR CMND            DUPLICATE STACK LOCATIONS
        LDA =FIXD
        JSR CMND            CONVERT POSITION TO AN INTEGER
        LDA =PTOF
        JSR CMND            DUPLICATE IT
        LDY =TEMP           WORK AREA
        STX XTEMP
        JSR POP
        DEY
        LDA (BASE),Y        GET THE HEX RESULT
        STA DVDN+1          AND PLACE FOR HEX-TO-BCD CONVERSION
        DEY
        LDA (BASE),Y
        STA DVDN
*
*            ROUTINE TO CONVERT A TWO-BYTE HEX NUMBER TO A FIVE-BYTE BCD
*            NUMBER.
*
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        LDX =4          COUNT OF FOUR
*
*           DIVIDE BY TEN
*
UNSPD   LDA =0
        STA RMNDR       CLEAR REMAINDER
        LDY =17         SET UP COUNT
        JMP D01
D02     LDA RMNDR
        SEC             SET CARRY FOR  SUBTRACT
        SBC DVSR
        BPL NREST       GO IF NO RESTORE
D01     CLC
        JMP MERGQ       GO TO SET Q
NREST   STA RMNDR       NEW RESIDUE
        SEC             Q=1
MERGQ   ROL DVDN+1
        ROL DVDN
        DEY             DECREMENT COUNT
        BEQ RTN
        ROL RMNDR       SHIFT LEFT
        JMP D02         CONTINUE
*
RTN     LDA RMNDR       GET REMAINDER
        STA DRES,X      STORE IT
        DEX
        BPL UNSPD       DO UNTIL DONE
*
        LDX XTEMP
        RTS
*
*
TOBCD2  JSR INTG        GET THE INTEGER PART
*
*           STORE THE POSITION COORDINATES ON VIDEO SCREEN
*
        LDY VY          GET VIDEO LOCATION
        LDA DRES+3      LOAD MSB
        ORA =$30        CHANGE TO ASCII
        STA $2200,Y
        INY
        LDA DRES+4      GET LSB
        ORA =$30
        STA $2200,Y
        INY
        INY
        STY VY
*
        LDA DRES+3
        ASL A           MOVE LOWER FOUR BITS TO UPPER FOUR BITS
        ASL A
        ASL A
        ASL A
        STA LAT,X       STORE THE DIGIT IN POSITION FIELD
        LDA DRES+4      GET NEXT DIGIT
        ORA LAT,X       MERGE WITH UPPER DIGIT
        STA LAT,X       REPLACE
*       INX
        CPX XLIM        SEE IF DONE
        BEQ OUT2        RETURN IF DONE
        LDA =FLTD
        JSR CMND        CHANGE INTEGER TO FLOATING-POINT
        LDA =FSUB
        JSR CMND        SUBTRACT OFF INTEGER PART
        LDY =C60        GET CONSTANT "60"
        STX XTEMP
        JSR PUSH
        LDX XTEMP
        LDA =FMUL
```

```
       JSR CMND        MULTIPLY RESIDUE BY 60
       JMP TOBCD2
OUT2   RTS
*
*
*****************************************************************
*                                                               *
*      ROUTINE CONVERTS RANGE AND BEARING FROM BINARY TO BCD    *
*      AND STORES THEM.                                         *
*                                                               *
*****************************************************************
*
RNGB   JSR INTG        GET THE INTEGER PART
*
*          VIDEO OUTPUT
*
       LDY VY
       LDA DRES+2
       ORA =$30
       STA $2200,Y
       INY
       LDA DRES+3
       ORA =$30
       STA $2200,Y
       INY
       LDA DRES+4
       ORA =$30
       STA $2200,Y
       INY
       INY
       STY VY
*
       LDA DRES+2      3RD ORDER DIGIT
       ASL A           MOVE LOWER DIGIT TO UPPER
       ASL A
       ASL A
       ASL A
       STA RHO,X       PUT IT IN RESULT
       LDA DRES+3      2ND ORDER DIGIT
       ORA RHO,X       MERGE WITH LAST DIGIT
       STA RHO,X
       INX
       LDA DRES+4      FIRST DIGIT
       ASL A
       ASL A
       ASL A
       ASL A
       STA RHO,X       STORE IT
       LDA =FLTD
       JSR CMND        CHANGE TO FLOATING-POINT
       LDA =FSUB
       JSR CMND        SUBTRACT OFF INTEGER
       LDY =C256       "256"
       STX XTEMP
       JSR PUSH
       LDA =FMUL
       JSR CMND        GET FRACTION AS INTEGER
       LDA =FIXD
       JSR CMND
       LDY =TEMP       WORK AREA
       JSR POP
       DEY
       LDA (BASE),Y    GET THE FRACTION
       LSR A           SEARCH-TABLE METHOD OF
       LSR A           OBTAINING CORRESPONDING
       LSR A           DECIMAL VALUE
       LSR A
       TAX             USE AS INDEX
       LDA FRTBL1,X    GET THE DECIMAL EQUIVALENT
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        LDX XTEMP
        LDY VY
        ORA =$30
        STA $2200,Y
        AND =$F
        ORA RHO,X       STORE THE FRACTION
        STA RHO,X
        RTS
*
*
*************************************************************************
*                                                                       *
*         CONSTANT TABLE OF NUMBERS USED IN CALCULATIONS.               *
*                                                                       *
*************************************************************************
*
TABLE   ADR (CNTS)
CNTS    EQU *
        HEX 7C,9F,BE,77   00 - TCY
        HEX 7C,DD,2F,1B   04 - TCZ
        HEX 7E,9A,EC,71   08 - THMY
        HEX 7E,98,0F,39   0C - THMZ
        HEX 06,BB,F2,6E   10 - XNR
        HEX 00,FD,13,63   14 - CTMY
        HEX 7E,9A,55,50   18 - STMY
        HEX 00,FD,2E,C3   1C - CTMZ
        HEX 7E,97,80,51   20 - STMZ
        HEX 7F,BF,D3,EB   24 - CXK
        HEX 00,ED,5A,69   28 - SXK
        HEX 02,BE,EC,DD   2C - C1
        HEX 03,89,F5,17   30 - C2
        HEX 83,E5,7E,A9   34 - C3
        HEX 03,82,DE,4C   38 - C4
        HEX 83,8E,DC,13   3C - C5
        HEX 01,93,27,AA   40 - C6
        HEX 83,A1,A2,81   44 - C7
        HEX 03,A3,CA,CE   48 - C8
        HEX 00,8F,28,B3   4C - C9
        HEX 77,CF,C0,08 50 - C10
        HEX 75,87,75,21 54 - C11
        HEX F8,C2,1E,A6 58 - C12
        HEX 01,80,6F,75 5C - C14
        HEX 69,D6,BF,95 60 - 1E-7
        HEX 0C,D6,F5,7B   64 - RCR
        HEX 02,80,00,00   68 - TWO ("2")
        HEX 09,80,00,00   6C - C256 ("256")
        HEX 06,E5,2E,E1   70 - P180 (180/PI)
        HEX 06,F0,00,00   74 - C60 (60)
*
*           END OF NUMBER TABLE; START SCRATCH SPACE
*
FRTBL1  HEX 00,01,01,02,03,03,04,04
        HEX 05,06,06,07,08,08,09,09
LSCRN   ASC 'LATLONGRNGBRNG..'
SCLC    HEX 68,69,6A
        HEX 88,89,8A,8B
        HEX A8,A9,AA
        HEX C8,C9,CA,CB
        HEX B3,D3
*
*
        END
```

Appendix IV. Program listing for program to compute spherical C-constant from equations 3-22 through 3-36. This program was written in FORTRAN-IV using double precision arithmetic.

ORIGINAL PAGE IS OF POOR QUALITY

```
C     THIS PROGRAM IS USED TO CALCULATE THE CONSTANTS USED IN THE
C     EXPLICIT COORDINATE CONVERTER TO CONVERT THE GEOCENTRIC
C     ARC ANGLES TO GEOCENTRIC COORDINATES (LAT. AND LONG.)
C     REFERENCE H. FELL, NAVIGATION, SUMMER 1975 AND PRIVATE
C     COMMUNICATION.
C
C     FIRST WRITTEN 11/1980
C     MAJOR REVISION 6/1981
C     J. P. FISCHER
C
      IMPLICITREAL*8(A-H,O-Z)
      DATA RR/6378.2064/,R/6356.5838/
      COMMON XX,YY,ZZ
      PRINT23
      CALL RDLL$(PHIO,THETO,&9)
C
C     CALCULATION OF THE CENTER OF T   OSCULATING SPHERE.
C
      B1=DSIN(PHIO)
      B2=B1*B1
      B3=DCOS(PHIO)
      B4=B3*B3
      B5=RR*RR
      B6=R*R
      RC=(B5*B2+B6*B4)/R
      B7=DSQRT(B5*B2+B6*B4)
      XX=(RR-B7)*B3*DSIN(THETO)
      YY=(RR-B7)*DCOS(THETO)*B3
      ZZ=(R-RR/R*B7)*B1
      C10=XX/RC
      C11=YY/RC
      C12=ZZ/RC
      C13=C10
      C14=RR/R
C
C     READ IN THE COORDINATES OF THE THREE LORAN-C STATIONS.
C
      CALL RDLL$(PM,TM,&9)
      CALL RDLL$(PX,TX,&9)
      CALL RDLL$(PY,TY,&9)
C
C     CONVERT THE GEODETIC LATITUDE TO GEOCENTRIC AND MAP THE POINTS
C     ONTO THE OSCULATING SPHERE.
C
      CALL SPD$(PM,TM,PHM,THM)
      CALL SPD$(PX,TX,PHX,THX)
      CALL SPD$(PY,TY,PHY,THY)
C
C     COMPUTE THE C-CONSTANTS USED TO CONVERT GEOCENTRIC ARC ANGLES
C     TO GEOCENTRIC LATITUDE AND LONGITUDE.
C
      A1=DCOS(PHM)
      A2=DSIN(PHM)
      A3=DCOS(THM)
      A4=DSIN(THM)
      A5=DCOS(PHX)
      A6=DSIN(PHX)
      A7=DCOS(THX)
      A8=DSIN(THX)
      A9=DCOS(PHY)
      A10=DSIN(PHY)
      A11=DCOS(THY)
      A12=DSIN(THY)
C
      D=A6*A1*A9*(A3*A12-A4*A11)-A2*A5*A9*(A7*A12-A8*A11)+A10*A5*A1*(A7*
     >A4-A8*A3)
      C1=(A1*A4*A10-A2*A9*A12)/D
      C2=(A6*A9*A12-A5*A8*A10)/D
      C3=(A5*A8*A2-A6*A1*A4)/D
      C4=(A2*A9*A11-A1*A3*A10)/D
      C5=(A5*A7*A10-A6*A9*A11)/D
      C6=(A6*A1*A3-A5*A7*A2)/D
```

```
      C7=A1*A9*(A3*A12-A4*A11)/D
      C8=A5*A9*(A8*A11-A7*A12)/D
      C9=A5*A1*(A7*A4-A8*A3)/D
C
C     PRINT THE RESULTS AND QUIT.
C
      PRINT1,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14
    9 STOP
    1 FORMAT(' C1 = ',1PD14.7,3X,'C2 = ',D14.7,3X,'C3 = ',D14.7/' C4 = '
     >,D14.7,3X,'C5 = ',D14.7,3X,'C6 = ',D14.7/' C7 = ',D14.7,3X,'C8 = '
     >,D14.7,3X,'C9 = ',D14.7/' C10 = ',D14.7,3X,'C11 = ',D14.7,3X,'C12
     >= ',D14.7/' C13 = ',D14.7,3X,'C14 = ',D14.7)
   23 FORMAT('1ENTER COORDINATES OF MIDPOINT OF COVERAGE AREA.')
      END
      SUBROUTINE SPDS(PHI0,THE0,PHI1,THE1)
C
C     THIS SUBROUTINE CALCULATES THE COORDINATES OF A POINT ON
C     A SPHEROID AND CALCULATES THE IMAGE OF P UNDER THE SPHEROID
C     TO THE OSCULATING SPHERE WITH CENTER AT XX, YY, ZZ FROM
C     MAIN PROGRAM.
C
      IMPLICIT REAL*8(A-H,O-Z)
      DATA RR/6378.2064/,R/6356.5838/
      COMMON XX,YY,ZZ
      PHI0=DATAN(R/RR*DTAN(PHI0))
      X=RR*DCOS(PHI0)*DSIN(THE0)
      Y=RR*DCOS(PHI0)*DCOS(THE0)
      Z=R*DSIN(PHI0)
      XL=DSQRT((X-XX)*(X-XX)+(Y-YY)*(Y-YY)+(Z-ZZ)*(Z-ZZ))
      PHI1=DARSIN((Z-ZZ)/XL)
      THE1=DATAN((X-XX)/(Y-YY))
      RETURN
      END
C
C     THIS SUBROUTINE CONVERTS GEOCENTRIC COORDINATES ENTERED BY THE
C     USER TO RADIAN COORDINATES.  INPUT FORM IS:  DDDD MM SS.SS
```

Appendix V. Program listings for service routine to convert BCD numbers entered at user's terminal to 9511 equivalent binary floating-point numbers. First program is written in FORTRAN-IV and second program is written in IBM 360/370 assembly language.

```
C     WHERE 'DDDD' IS THE DEGREES PORTION OF THE LAT. OR LONG., INCLUDING
C     SIGN, MM IS THE MINUTES PORTION, AND SS.SS IS THE SECONDS PORTION.
C     READ FORMAT IS:  I4,1X,I2,1X,F5.0.
C
      IMPLICIT REAL*8(A-H,O-Z)
      DATA PI/3.1415926535898/
      DATA MSG1/'LATI'/,MSG2/'TUDE'/,MSG3/':   '/,MSG4/'LONG'/,MSG5/'ITU
     >D'/,MSG6/'E:  '/
      PI1=PI/180.
C
C        PROMPT USER.
C
      PRINT1,MSG1,MSG2,MSG3
      READ(5,10,END=21)T1,T2,T3
      PHI=PI1*(T1+(T2+T3/60.)/60.)
C
C        PROMPT USER FOR LONGITUDE ENTRY.
C
      PRINT1,MSG4,MSG5,MSG6
      READ(5,10,END=21)T1,T2,T3
      THE=PI1*(T1+(T2+T3/60.)/60.)
C
      RETURN
   21 RETURN 1
    1 FORMAT(' ENTER ',3A4/'  DDDD MM SS.SS')
   10 FORMAT(F4.0,1X,F2.0,1X,F5.0)
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C     FORTRAN PROGRAM 9511
C     THIS PROGRAM IS DESIGNED TO RUN IN CONJUCTION WITH SUBROUTINE
C     DECHX.  A REAL NUMBER IS READ IN AND CONVERTED FROM 360/370
C     FLOATING-POINT FORMAT TO THE BINARY FLOATING-POINT FORMAT
C     USED BY THE AM9511A MATH CHIP.  THE INPUT NUMBER IS HANDLED
C     AS REAL*8 TO ALLOW ROUNDING UP OF THE FINAL RESULT.
C
C     J.P. FISCHER     4/81
C
      REAL*8 XA
      LOGICAL*1 R1,R2,R3,R4
    1 READ(5,10,END=2)XA
      CALL DECHX(XA,R1,R2,R3,R4)
      PRINT 11,R1,R2,R3,R4
      GOTO1
    2 STOP
   10 FORMAT(F15.0)
   11 FORMAT(1X,Z2,1X,3Z2)
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
************************************************************************
*                                                                      *
*        THIS FORTRAN-COMPATIBLE SUBROUTINE TRANSLATES A 360/370       *
*        FLOATING-POINT NUMBER TO THE FLOATING-POINT FORMAT USED       *
*        BY THE AM9511 MATH CHIP.  THIS IS DESIGNED AS A DEVELOPMENT   *
*        PROGRAM.  RESULTING NUMBER IS RETURNED AS FOUR SEPARATE       *
*        NUMBERS (LOGICAL*1).                                          *
*                                                                      *
*        J. P. FISCHER    3/81                                         *
*                                                                      *
************************************************************************
         SPACE 2
DECHX    CSECT
         BC    15,5+1+4(15)   BRANCH AROUND ID
         DC    XL1'5'
         DC    CL5'DECHX'
         LR    12,15          NEW BASE
         USING DECHX,12
         STM   14,12,12(13)   SAVE REGISTERS
         SPACE
         MVI   FLAG,0         CLEAR FLAG
         L     2,0(1)         ADDRESS OF FLOATING-POINT NUMBER
         MVC   EXPO(5),0(2)   MOVE INTO THIS PROGRAM
         TM    EXPO,X'80'     SEE IF MANTISSA IS NEGATIVE
         BZ    POSM           GO IF POSITIVE
         MVI   FLAG,1         SET MINUS BIT
         NI    EXPO,X'7F'     CLEAR NEGATIVE BIT
POSM     SLR   3,3            ZERO A REGISTER
         IC    3,EXPO         GET THE EXPONENT
         SL    3,F64          SUBTRACT X'40' FOR NEW BINARY EXPONENT
         SLL   3,2            CONVERT EXPONENT FROM HEX TO BINARY
         SPACE
LEFT     TM    EXPO+1,X'80'   SEE IF LEFT ADJUSTMENT NECESSARY
         BO    NOT            IF SET, DON'T DO ANY MORE
         ICM   4,15,EXPO+1    GET THE MANTISSA
         SLL   4,1            SHIFT OVER ONE
         BCTR  3,0            DECREMENT EXPONENT
         STCM  4,15,EXPO+1
         B     LEFT           SEE IF SHIFTING COMPLETE
NOT      TM    EXPO+4,X'80'   SEE IF THIS BIT IS SET
         BZ    POS            IF NOT, DON'T ROUND UP
         ICM   4,B'0111',EXPO+1    GET THE MANTISSA
         LA    4,1(4)         INCREMENT IT
         STCM  4,B'0111',EXPO+1    REPLACE IT
         SPACE
POS      STC   3,EXPO         STORE THE NEW EXPONENT
         TM    FLAG,1         SEE IF NEGATIVE
         BZ    POS1
         OI    EXPO,X'80'     SET THE NEGATIVE BIT
         B     POS2
```

```
POS1     NI    EXPO,X'7F'      CLEAR THE NEGATIVE BIT
         SPACE
*              RETURN THE NEW FLOATING-POINT NUMBER
         SPACE
POS2     LM    3,6,4(1)        ADDRESSES OF RETURN LOCATIONS
         MVC   0(1,3),EXPO     MOVE THE NEW EXPONENT
         MVC   0(1,4),EXPO+1   MOVE THE MSB OF MANTISSA
         MVC   0(1,5),EXPO+2   MOVE THE MIDDLE OF MANTISSA
         MVC   0(1,6),EXPO+3   MOVE THE LSB OF MANTISSA
         SPACE
*              RETURN TO CALLER
         SPACE
         LM    2,12,28(13)     RESTORE REGISTERS AS THEY WERE
         MVI   12(13),X'FF'    INDICATE CONTROL RETURNED TO CALLER
         SR    15,15           ZERO RETURN CODE
         BR    14              RETURN
         SPACE 3
         DS    0F
F64      DC    F'64'
EXPO     DS    XL5
FLAG     DS    XL1
         END   DECHX
```