

FOR REFERENCE

NOT TO BE TAKEN FROM THIS ROOM

MICROPROCESSOR BASED
DEAD RECKONING TABLE

by

Tayfun EVREN

Submitted to the Faculty of the School
of Engineering in Partial Fulfillment of
the Requirements for the Degree of

MASTER OF SCIENCE

in

ELECTRICAL ENGINEERING

Bogazici University Library



39001100315905

14

Boğaziçi University

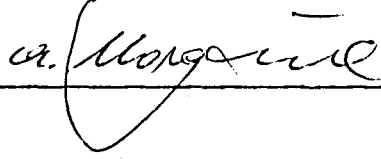
1983

This thesis has been approved

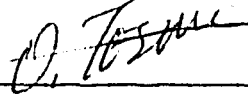
Yard.Doç.Dr. Ömer CERİD
(Thesis Supervisor)

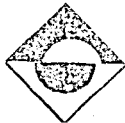


Yard.Doç.Dr. Avni MORGÜL



Yard.Doç.Dr. Oğuz TOSUN





ACKNOWLEDGEMENT

I would particularly like to thank my thesis supervisor, Dr. Ömer Cerid, for his useful comments and guidance.

I am also grateful to Mehmet Temizyürek for his friendly support.

I would like to thank my wife for her careful typing and patience.

ABSTRACT

The application areas of microprocessors have been growing recently. In addition to various areas, efficient and reliable results have been obtained in military applications.

The object of this thesis is to apply microprocessor control on a device used in the Navy and the realization of the necessary hardware and software and the observation of the results.

In this study, the Dead Reckoning Table used on the ships of the Navy was controlled and operated by microprocessor. The Dead Reckoning Table is used for recording the movements of our ship and the others in the same operation area. Many tactical and manoeuvre problems are solved on this table. The preparing speed and the reliability of the records are increased and the operator mistakes are prevented with the aid of microprocessor.

ÖZET

Son yıllarda mikrobilgisayarların uygulama sahaları hızla artmaktadır. Çeşitli kullanım yerlerine ek olarak askeri uygulamalarda da verimli ve güvenilir neticeler elde edilmektedir.

Bu tezin amacı mikroişlemcinin Deniz Kuvvetlerinin savaş gemilerinde kullanılmakta olan bir aygıta uygulanabilirliğine karar vermek ve gerekli donanım ve yazılımı gerçekleştirerek sonuçları incelemektir.

Yapılan çalışmada Deniz Kuvvetlerine ait gemilerin Savaş Harekat Merkezlerinde bulunan iz masalarının mikroişlemci ile kontrolü ve çalışması gerçekleştirilmiştir. İz masasının görevi kendi gemimiz ile birlikte hareket sırasında bulunan diğer gemilerin bütün hareketlerini kağıt üzerine işaretlemek, taktik ve manevra ile ilgili problemlerin çözülmesine yardımcı olmaktır. Mikroişlemci kullanılması ile kayıtların tutulmasında sürat ve güvenilirlik sağlanarak operatör hataları önlenmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

ABSTRACT

ÖZET

CHAPTER 1	INTRODUCTION	1
CHAPTER 2	M6800 MICROCOMPUTER SYSTEM	4
	System Components	
	Microprocessing Unit	
	Memory	
	Input/Output Interface	
CHAPTER 3	DESCRIPTION OF THE MICROPROCESSOR BASED DEAD RECKONING TABLE	14
	Microprocessor Based System Design	
	System Layout	
CHAPTER 4	INTERFACE CIRCUITS	18
	Input Circuits	
	Output Circuits	
	Timing Circuit	
CHAPTER 5	SOFTWARE DEVELOPMENT	31
	Main Program	
	Interrupt Service Routine	
CHAPTER 6	DISCUSSION	65
CHAPTER 7	CONCLUSION	66

BIBLIOGRAPHY	67
APPENDIX A PROGRAM LISTING	69
APPENDIX B MC 6800 INSTRUCTION SET	85
APPENDIX C COST	87
APPENDIX D PRINTED CIRCUIT BOARDS	88

CHAPTER I

INTRODUCTION

The Dead Reckoning Table (DRT) is used at the combat Information Centers (CIC) of the Naval ships. Most of the operation records in an exercise are prepared on this table.

The DRT is a projection device with a transparent plastic top. A ship's position pointer which is driven by gears is projected through the plastic. A sheet of translucent paper, placed over the top and temporarily taped down at the corners, is marked by the operators with the position of the ship at regular intervals. This paper may be considered to be a map. The other ships in the same operation area can also be plotted. The geographical positions and movements are displayed on the paper. When the ship's track runs out of the edge of the paper the pointer is placed to another location and a new piece of paper is put on the table.

Many operation problems are solved on this paper. This paper is the summary of an operation at the sea.

The course and the speed of the ship are the inputs to DRT. These signals are transmitted from the gyro compass and the log as synchro signals. The ship's position pointer moves with these input signals and shows the position of the ship.

In order to plot one or more target ships on the same paper the operator finds out the bearing and the distance of them from his ship. Then he draws the bearing line in the length of the distance and plots the target's position. The operator also records the time at which the plotting is done. The time intervals between the marks is adjusted by the operator.

The purpose of this thesis is to design and build a microprocessor based DRT. Many improvements will be achieved by using the microprocessor control on the DRT. Primarily, these are:

- No personnel will be required for plotting.
- The ship's position pen will automatically shift to the opposite edge when running out of the paper.
- Scale changes can be made by push button.
- Remote read outs can be made available to the other parts of the ship.
- Modifications or improvements via software changes.

- Entry and display of one or more targets.
- Plotter can print the time on the plot.
- Automatic printing of the data.
- Automatic changing of the paper.
- Solutions of the manoeuvre and operation problems.

The small Fast Patrol Boats (FPB's) which have a limited number of seaman onboard move with a very high speed in the operation area. A microprocessor controlled DRT will be very useful for FPB's.

In this study most of the items above were realized, but some of them were considered as a further work.

In addition to the plotting of the ships, the input data and the calculated values such as ship's course and target speed are transmitted to a teletype-writer which displays the situation. This information can be delivered to the other departments to inform the personnel about the operation.

CHAPTER 2

M6800 MICROCOMPUTER SYSTEM

System Components

The microcomputer system consists of:

1. Microprocessing Unit (MPU)
2. Clock Generator
3. Read Only Memory (ROM)
4. Random Access Memory (RAM)
5. Parallel Input/Output Interface
6. Asynchronous Serial Input/Output Interface

Fig. 2-1 shows the block diagram of the microcomputer system.

This block diagram represents all of the hardware required for a fully operational microcomputer system.² The data bus is shared fully between all devices in the system. The control bus is shared by all devices which get their own required signal from the bus. Different combinations of signals may be received from the address bus to define where each device is located. The address bus selects one byte or location from 64 K bytes or locations.

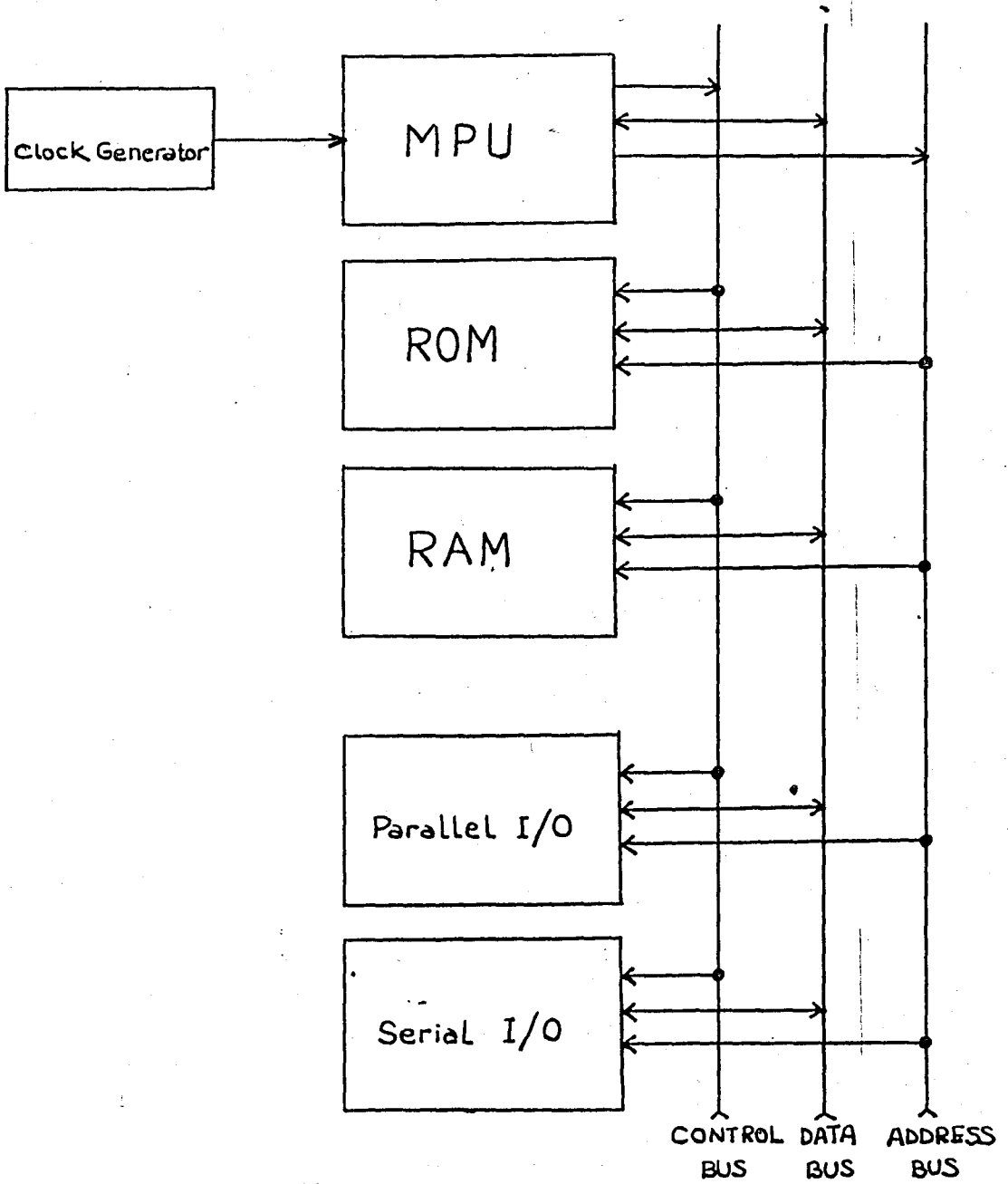


Fig.2-1 Block Diagram of the Microcomputer System.

Microprocessing Unit

MC6800 MPU is the heart of the system. It is a monolithic 8-bit microprocessor performing the central control function for the system. The MPU can execute 72 different instructions including arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, jump, interrupt and stack manipulation instructions.⁴ The MPU has three 16-bit registers and three 8-bit registers available for use by the programmer. These are shown in Fig.2-2. The program counter points to the current program address.

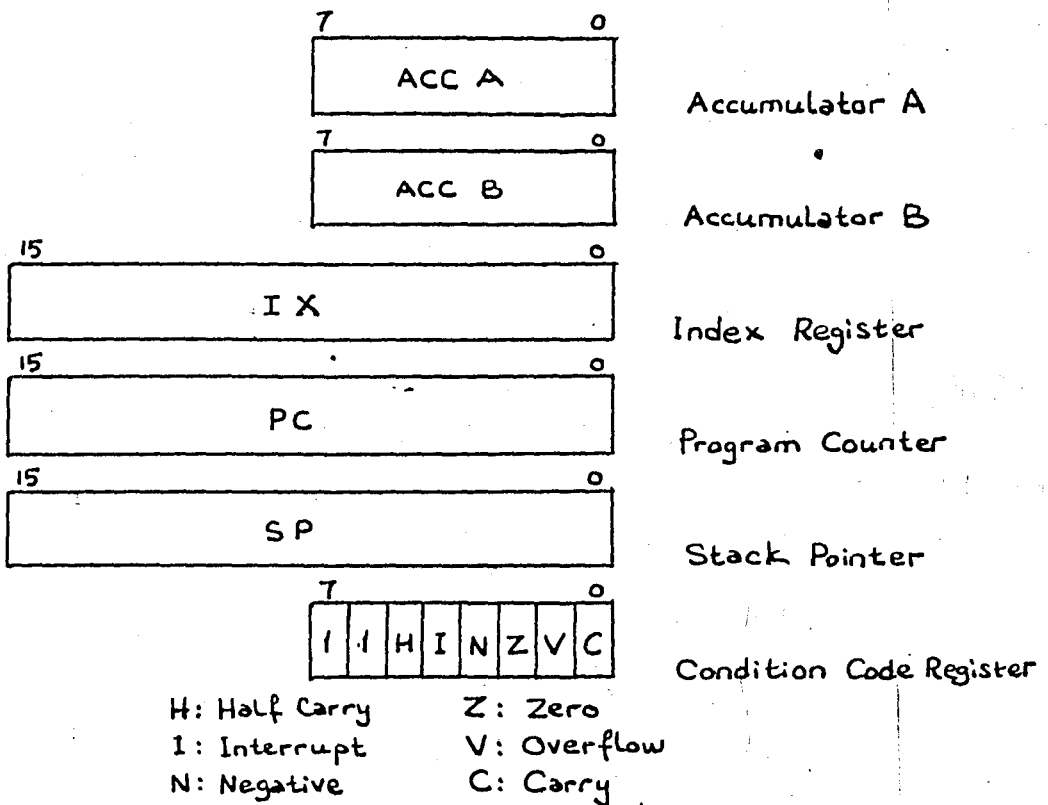


Fig.2-2 The Registers of MC 6800

The stack pointer contains the address of the next available location in the stack which is a RAM. Index register is used to store data or a sixteen bit memory address for the indexed mode of addressing. The two accumulators are used to hold operands and results from the arithmetic logic unit (ALU). The condition code register indicates the results of an ALU operation. The bits of this register are used as testable conditions for the conditional branch instructions. Bit 4 is the interrupt mask bit. The unused bits of the condition code register are ones.

A general block diagram of the MC 6800 is shown in Fig.2-3. The control and timing signals regulate the system operation.

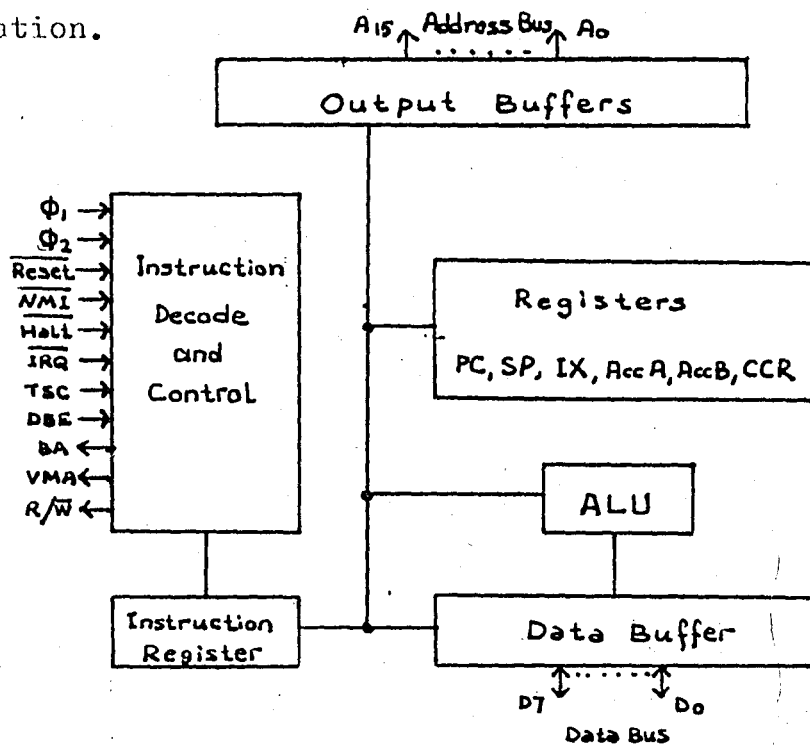


Fig.2-3 The Block Diagram of the MC 6800.

Φ_1 and Φ_2 are two-phase non-overlapping clock signals. The $\overline{\text{Halt}}$ input is used to stop the MPU. In the Halt mode all three-state lines will be in three-state mode. Three State Control (TSC) input causes all of the address lines and the $\overline{\text{R/W}}$ line to go into the high impedance state. $\overline{\text{R/W}}$ line signals the peripherals and memory devices whether the MPU is in Read or Write state.

Valid Memory Address (VMA) output indicates to peripheral devices that there is a valid address on the address bus. This signal is applied to the enable or chip select inputs of each family device in order to disable data transfer when VMA is low. Data Bus Enable (DBE) input is the three state control signal for the MPU data bus. Normally, this signal will be Φ_2 , derived from the clock. Bus Available (BA) signal will normally be in the low state; when activated, it will go to the high state indicating that the MPU has stopped. The interrupt request ($\overline{\text{IRQ}}$) is a level sensitive input which requests that an interrupt sequence be generated within the machine. If the interrupt mask bit in the CCR is not set, the machine will begin an interrupt sequence. A low-going edge on the Non-Maskable Interrupt ($\overline{\text{NMI}}$) input requests that a non-maskable interrupt sequence be generated within the processor. The interrupt mask bit in the CCR has no effect on $\overline{\text{NMI}}$. Reset input is used to reset and start

the MPU from a power down condition.

The MC 6800 has seven addressing modes that can be used by the programmer, with the addressing mode a function of both the type of the instruction and the coding within the instruction. In Accumulator Addressing mode, either accumulator A or accumulator B is specified. In Immediate Addressing, the operand is contained in the second byte of instruction except LDS and LDX instructions which have the operand in the second and third bytes of the instruction. In Direct Addressing mode, the address of the operand is contained in the second byte of the instruction. In Extended Addressing, the address contained in the second byte of the instruction is used as the higher eight bits of the address of the operand. The third byte of the instruction is used as the lower eight bits of the address for the operand. In Indexed Addressing, the operand of the instruction is added to the index register to provide the address. In Implied Addressing mode, the mnemonic operator specifies one or more registers which contain operands or in which results are saved. In Relative Addressing, the address contained in the second byte of the instruction is added to the program counter's lowest eight bits plus two. The carry or borrow is then added to the higher eight bits.

Memory

The ROM section is permanent data storage. The ROM keeps the data whether the power is on or not. The RAM section will keep its stored data as long as power is supplied to it and can be written into or read out of as desired. The ROM is used in order to storage the system start up instructions, monitor programming and all the necessary software. The RAM will be used for temporary data and program storage.

To read an information stored in the memory; the R/W line is placed in a read state, the address of the memory cell to be read is loaded onto the address lines, the memory is enabled by applying the proper signals to the chip enable lines, after a length of time required to propagate the information to the output, the data is read from the data out lines.

To write an information to the memory; the address at which the data to be stored is applied to the address bus, the memory is enabled, R/W line is placed in write state and the data to be written is applied to the data in lines.¹

Input / Output Interface

The MC 6821 Peripheral Interface Adapter (PIA) is used as parallel I/O interface. This device is capable of interfacing the MPU to peripherals through two 8-bit bidirectional peripheral data buses and four control lines. The simplified block diagram of the PIA is shown in Fig.2-4.

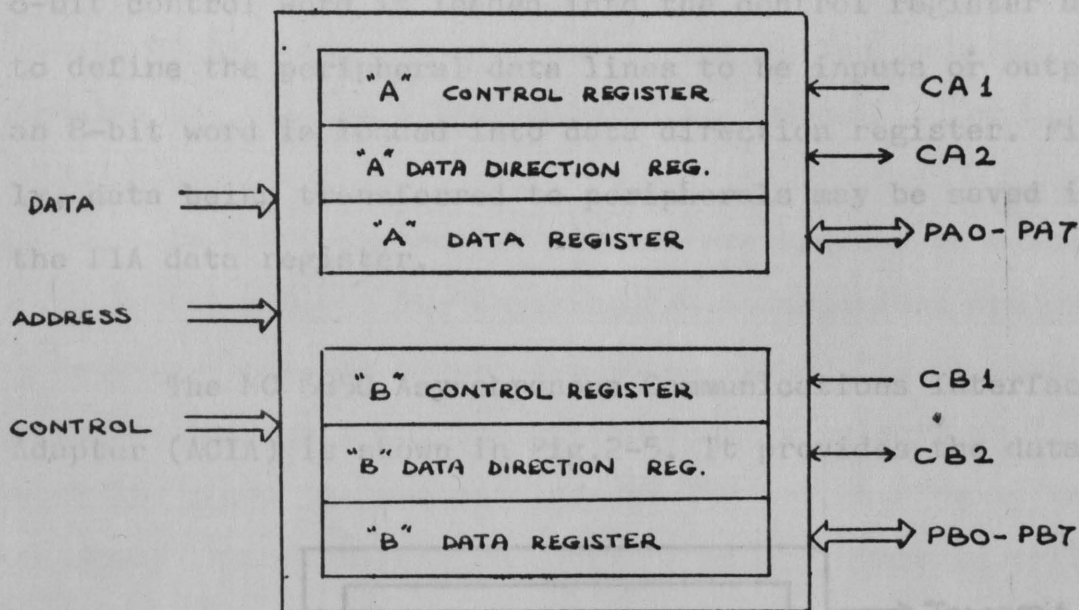


Fig. 2-4 The Simplified Block Diagram of the PIA.

The PIA is programmed by the MPU during system initialization. Each of the peripheral data lines can be programmed to act as an input or output and each of the four control

Fig. 2-5 The PIA Registers.

or interrupt lines may be programmed for one of the several control modes. The PIA is programmable in the sense that the MPU can read and/or write into its internal registers. There are six 8-bit registers in the PIA. They are separated into an A and B side, each side containing a Control Register, Data Direction Register and a Data Register. To define the operation of the PIA control lines, an 8-bit control word is loaded into the control register and to define the peripheral data lines to be inputs or outputs, an 8-bit word is loaded into data direction register. Finally, data being transferred to peripherals may be saved in the PIA data register.

The MC 6850 Asynchronous Communications Interface Adapter (ACIA) is shown in Fig.2-5. It provides the data

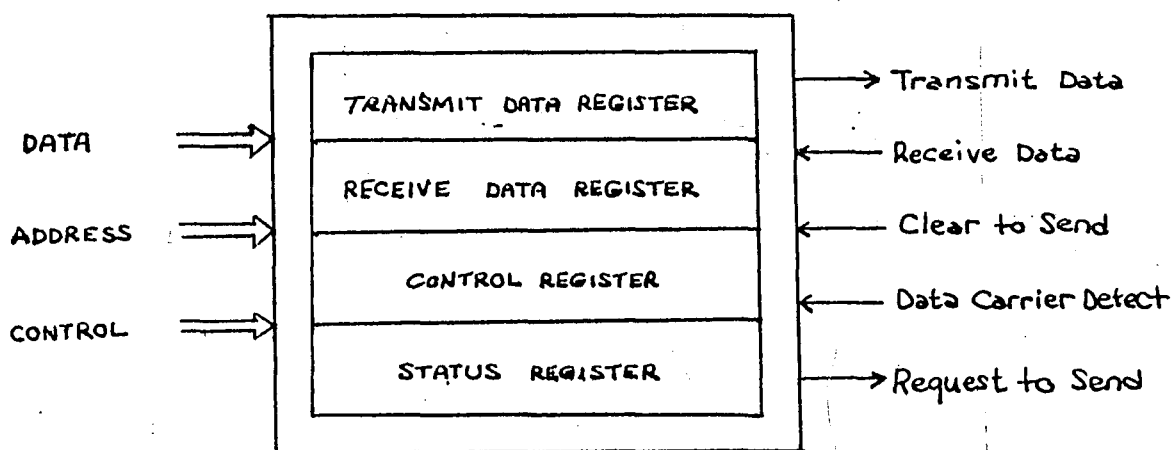


Fig. 2-5 The ACIA Registers.

formatting and control to interface serial asynchronous data communications information to bus organized systems. The parallel data of the bus is serially transmitted or received by the ACIA. The ACIA has four registers: Control, Status, Transmit Data and Receive Data.

The ACIA is programmed by loading an 8-bit word into the control register. The status and error conditions are monitored by reading the status register. The data buffers provide the storage of the data to be transmitted or received.

In M6800 system I/O devices are treated as a location in the memory. The separate I/O instructions are unnecessary since the memory can house either I/O or memory data in its locations. The microprocessor selects an I/O port and reads the contents of the I/O port buffer in the same way that data gets read out of memory.⁹ This is called "Memory Mapped I/O".

CHAPTER 3

DESCRIPTION OF THE MICROPROCESSOR BASED DRT

Microprocessor Based System Design

The development of a microprocessor based system contains some important steps. During development, the designer performs the following steps to satisfy the system requirements:

1. System design criteria: This step includes the consideration of the cost of hardware, software, designing effort, flexibility, compatibility, reliability, speed and availability.

2. The efficiency of the microprocessor than hardware logic.

3. Microprocessor selection criteria: In this step the designer takes the viability, the system and the software support and the upgrading potential of the microprocessor into account.

4. Designing the auxiliary hardware.

5. Software development.

6. Integrating the hardware and the software.

7. Debugging and testing.
8. Developing the maintenance facilities.

System Layout

The general block diagram of the system is shown in Fig. 3-1. The system is built around the M6800 Based Microcomputer because it is extensively supported in the university. All the necessary data inputs are simulated by developed circuits. The input and the output signals are received or transmitted by the 6821 PIA's.

The own course and the target bearing inputs are allowed to be eight-bit Gray code. The own ship speed and the target distance inputs are implemented as eight-bit binary signals. The binary word at the own speed input port corresponds the velocity of the ship in miles per hour (knots). The eight-bit binary number at the target distance input port is multiplied by 100 in order to find the real distance in yards.

For the reset input, two eight-bit ports are required. One each for the two bytes representing desired x and y position to the full plotter range of eight bits. The reset position is considered as a reference position for

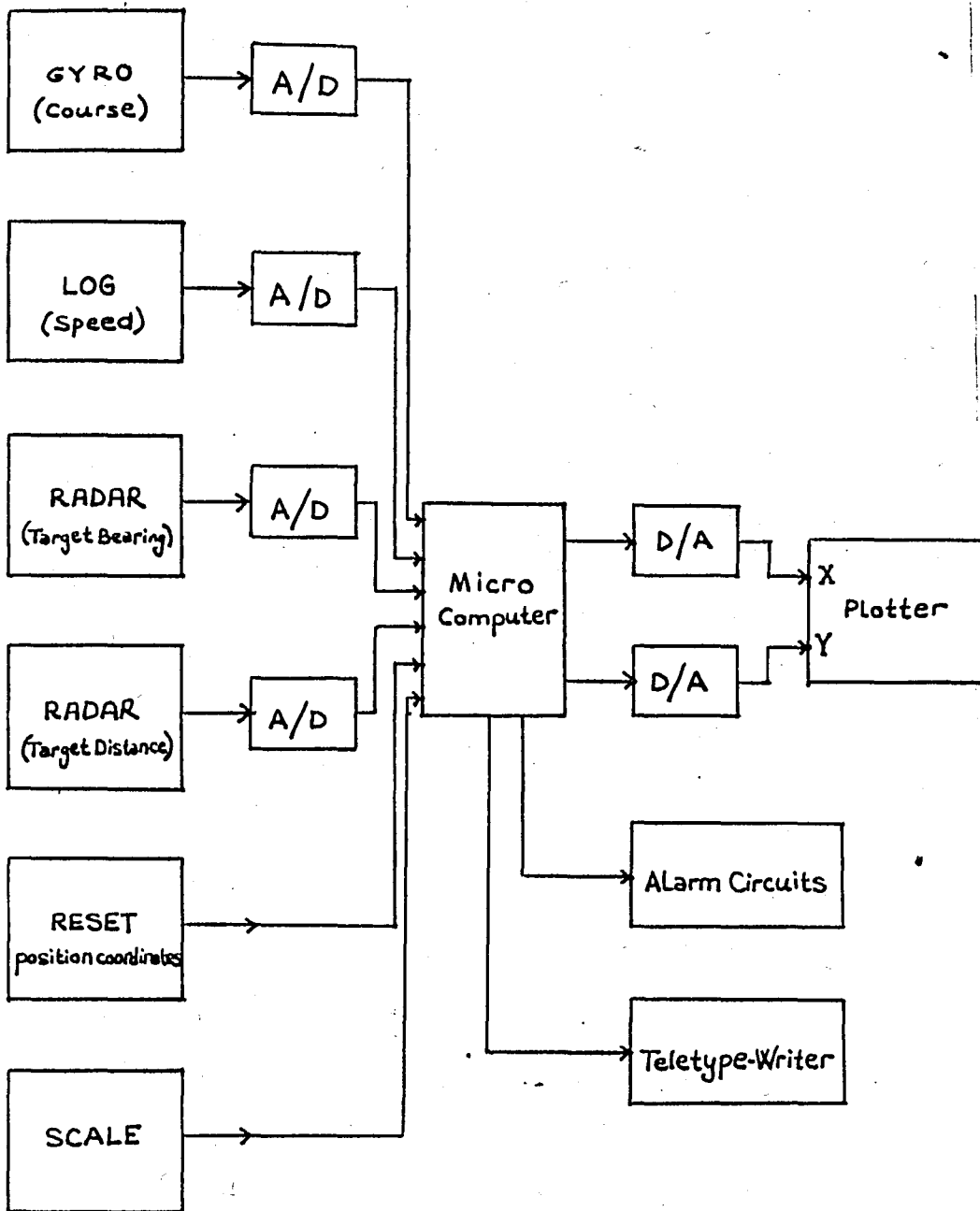


Fig. 3-1 Block Diagram of the Microprocessor Based DRT.

the plotting. This position should be chosen in according to the route of our ship. For example, the reference position should be at any desired place at north for a ship going southwards.

The sixteen-bit scale factor is required for the trigonometric calculations. The different scales can be used for various operations. Two additional ports are required to output the eight-bit x and eight-bit y coordinates to the plotter. The digital outputs are converted to analog by eight-bit digital to analog converters.

The positions of the target and own ship are marked with different figures on the plotter surface. If the positions mentioned above are outside of the plotting area, alarm circuits inform us about the situation. The data about the ships can be printed and displayed by sending the course and the speed information signals to the teletype-writer.

CHAPTER 4

INTERFACE CIRCUITS

Input Circuits

Own ship's course and the bearing of the target are transmitted from the gyro and the radar, respectively. They are synchro signals. In order to get the digital representation of the angle, the shaft encoder may be used.

The Gray code is useful for shaft encoders because the change of only 1 bit for each increment eliminates false intermediate codes that could occur in natural binary conversion. Gray code is closely related to binary in which the bit-position does not signify a numerical weighting. Gray code is a unit distance code. The unit distance property means that in going from the coding for any integer I to the coding for $I+1$ only one bit changes, regardless of the value of I . This property is usefully employed in transferring a coded variable reliably between a transducer and an instrument input.

The general structure of an n -bit Gray code can be described by the following scheme:

1. Except for beginning and ending of the code the least significant bit (LSB) g_0 alternates two zeros, two ones, two zeros etc., g_1 alternates four zeros, four ones, four zeros etc. and g_k alternates 2^{k+1} zeros, 2^{k+1} ones, 2^{k+1} zeros etc.

2. The first transition from 0 to 1 for bit g_{k+1} occurs right in the middle of the first string of ones for bit g_k .

The results of Gray code and natural binary code for optical shaft encoders with 4-bit resolution are illustrated in Fig.4-1. Note that, with the Gray code converter, there is only a 1-bit change at each transition. If the edge of a shaded area is slightly out of line, the coding will be in error by a small fraction of an LSB. In binary converter, however, all 4 bits change at once at the 180-degree and 360-degree transitions. If bit 2's shaded area were to end a little before from the 180-degree transition, the code, in a small region, would be 0011, indicating the 67.5 degree range. If the other shaded areas are out of line, there will be a total of 15 combinations of wrong range indication.

In this thesis an 8-bit shaft encoder was developed.

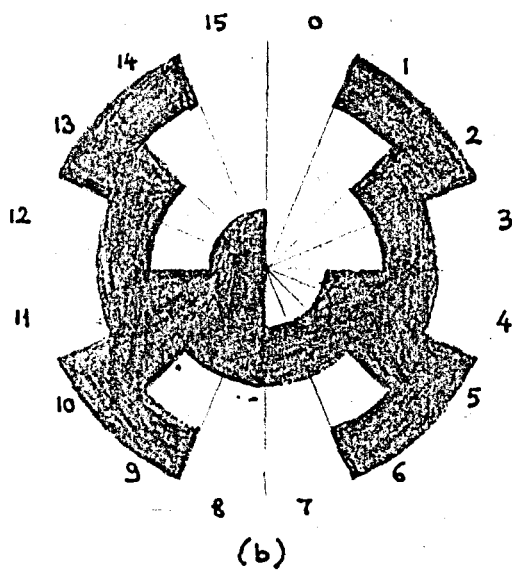
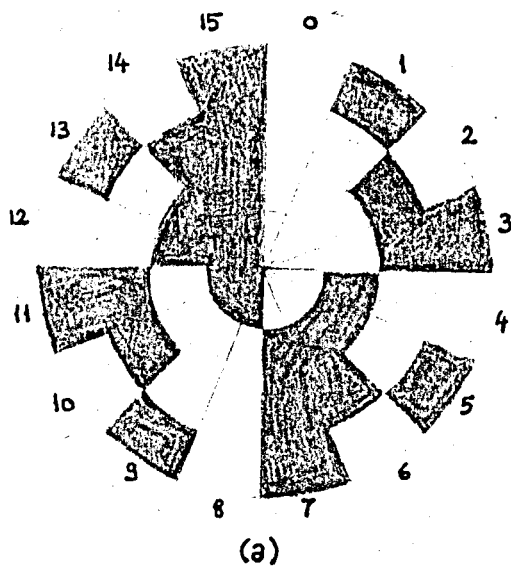


Fig.4-1. The results of Gray versus binary encoding. a) Binary encoding, b) Gray code encoding.

This encoder divides one revolution of its shaft into 256 parts. The angle of each part is $45/32$ degree as shown in Fig. 4-2.

In order to perform the calculations in the MPU, the Gray code is converted to the conventional binary code by software. The conversion from the Gray code to the binary code can be simply formulated as follows.⁶ Let

$$b = b_n b_{n-1} \dots b_2 b_1 b_0 \quad \text{conventional binary number and}$$

$$E = E_n E_{n-1} \dots E_2 E_1 E_0 \quad \text{Gray code equivalent}$$

Then

$$b_n = E_n$$

$$b_{n-j} = \bigoplus_{k=n}^{k=n-j} E_k = E_n \oplus E_{n-1} \oplus \dots \oplus E_{n-j}$$

where the operator \oplus signifies binary addition without carry, i.e., exclusive or operation. The binary MSB will be the same as the Gray code MSB. Then, continuing to read from MSB to LSB, if the next bit is 1, the next binary bit is the complement of the previous binary bit. For example, Gray code 0111000111 becomes 0101111010 in binary as shown below:

Gray code	0 1 1 1 0 0 0 1 1 1
Binary	0 → 1 → 0 → 1 → 1 → 1 → 1 → 0 → 1 → 0

The coding wheel in Fig. 4-2 is used in an instrument as shown in Fig. 4-3. The output signal of this device is own ship course input to the system.

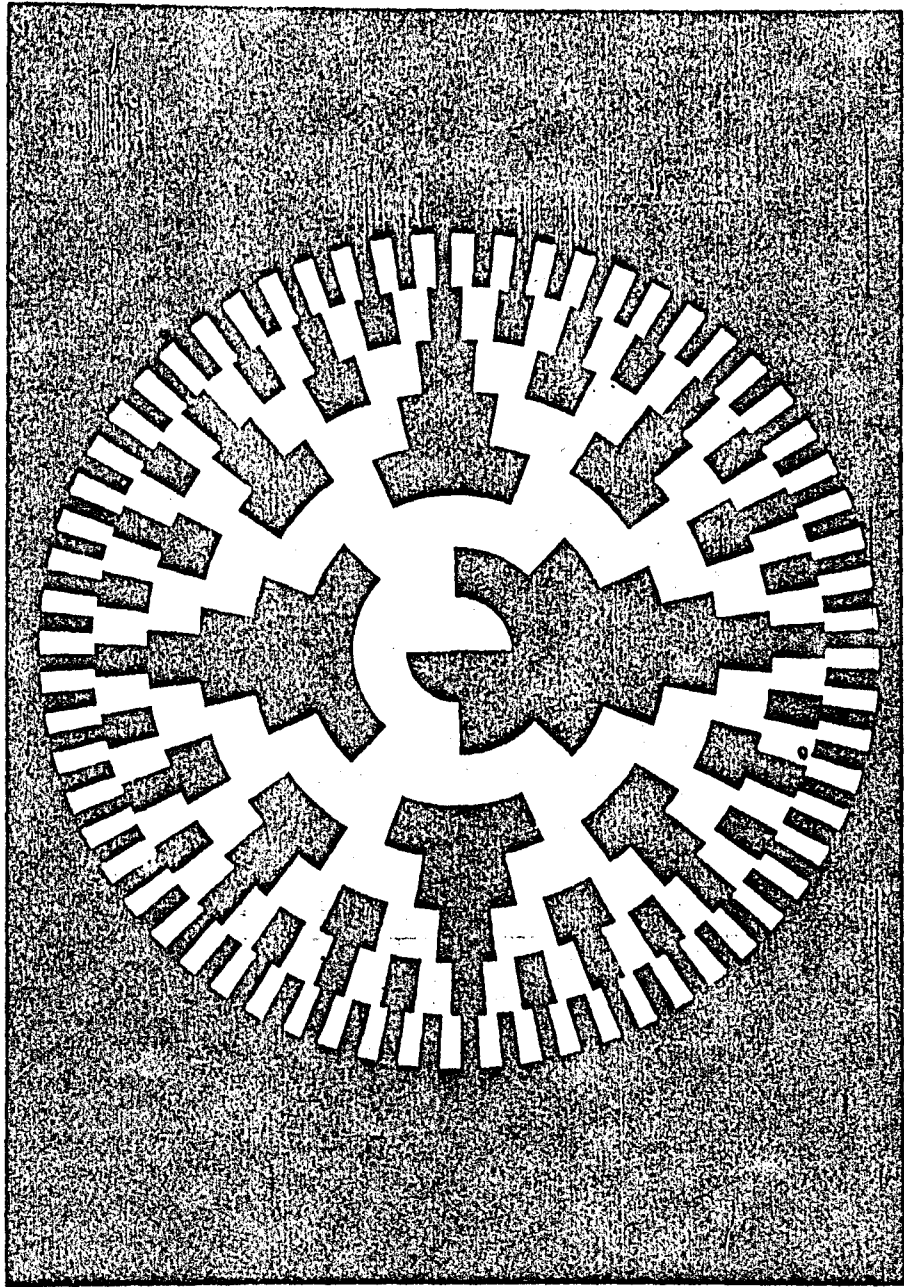


Fig.4-2. Eight-bit Optical Shaft Encoder.

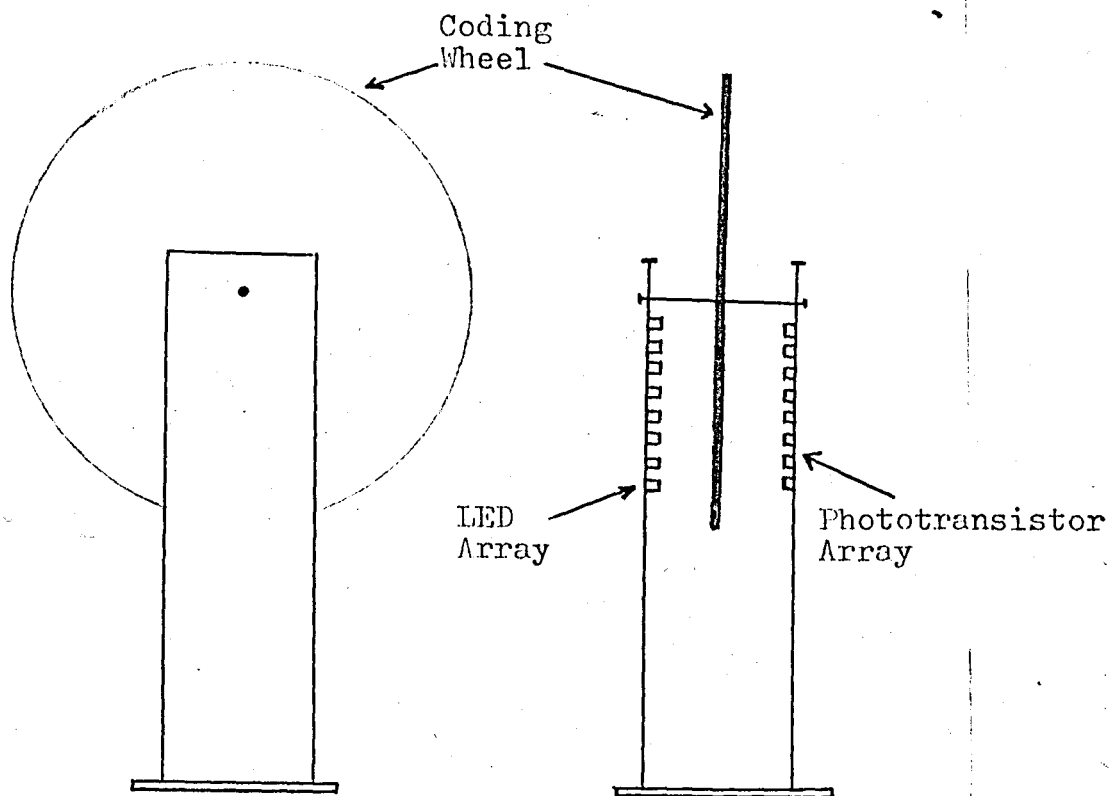


Fig. 4-3. Shaft Encoder Instrument.

The position of the wheel is detected by the phototransistors with the circuit in Fig.4-4.

Some current flows through the illuminated phototransistor. The voltage across the emitter resistor is compared with a reference level by the LM 339. The TTL level output of the comparator IC is fed to the PIA port programmed as input. The adjustment of the shaft encoder can be made by the 47 K potentiometers.

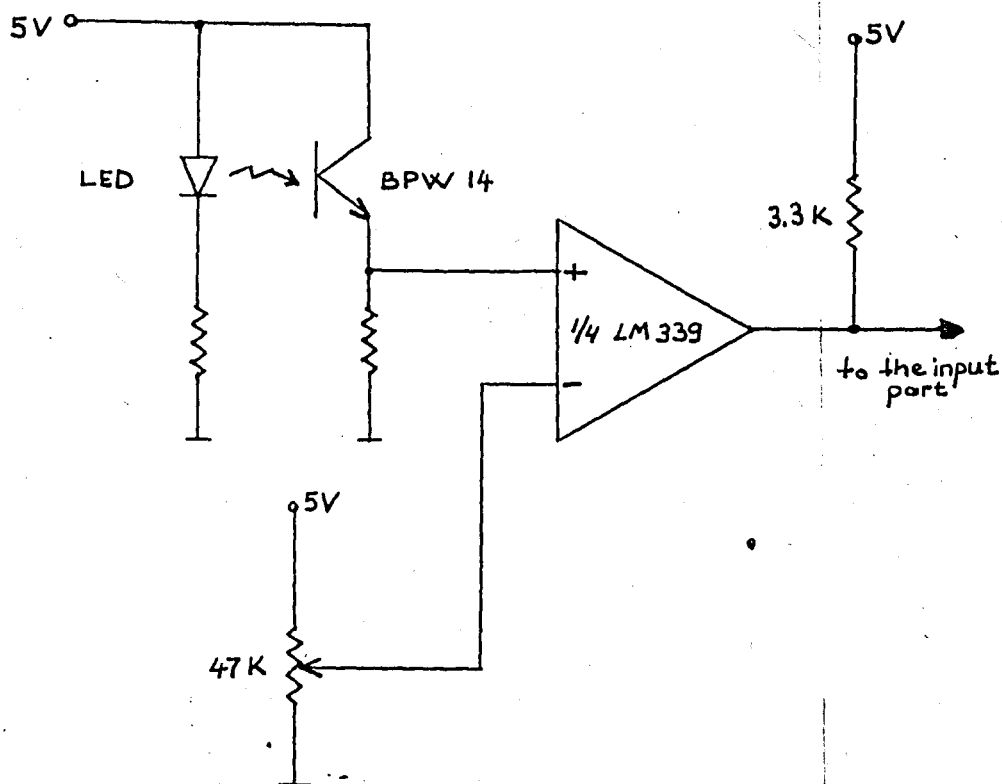


Fig. 4-4 A LED-Phototransistor Pair from the Array.

The other inputs to the system are; own ship's speed, target bearing, target distance, reference coordinates and the scale factor. These input signals are realized by switches except the scale factor. The scale factor is placed in a memory location because there is not one more port in the

system. Each of the signals above occupies an 8-bit port of the PIA's. The target bearing signal is accepted as Gray coded. It is converted to the binary code by software.

Output Circuits

The position coordinates calculated by the micro-computer are transmitted via the 6821 PIA are in 8-bit digital form. A and B ports of the PIA hold the X and Y coordinates respectively. The 8-bit digital data are converted to analog form by MC 1408L8 digital-to-analog converter. The output current of the MC 1408L8 is a linear product of an 8-bit digital word and an analog reference voltage. The 741 operational amplifier is used to obtain the voltage level in proportion to the analog current of the output of the D-to-A converter. Fig.4-5 shows the D-to-A converter circuit.

There are two digital to analog converter circuits, one for the horizontal axis and the other for the vertical axis. The output voltages of the operational amplifiers are applied to the inputs of the XY Plotter.

The voltage level at the output of the D to A converter is chosen as 1 volt when the digital word is FF in

hexadecimal notation.

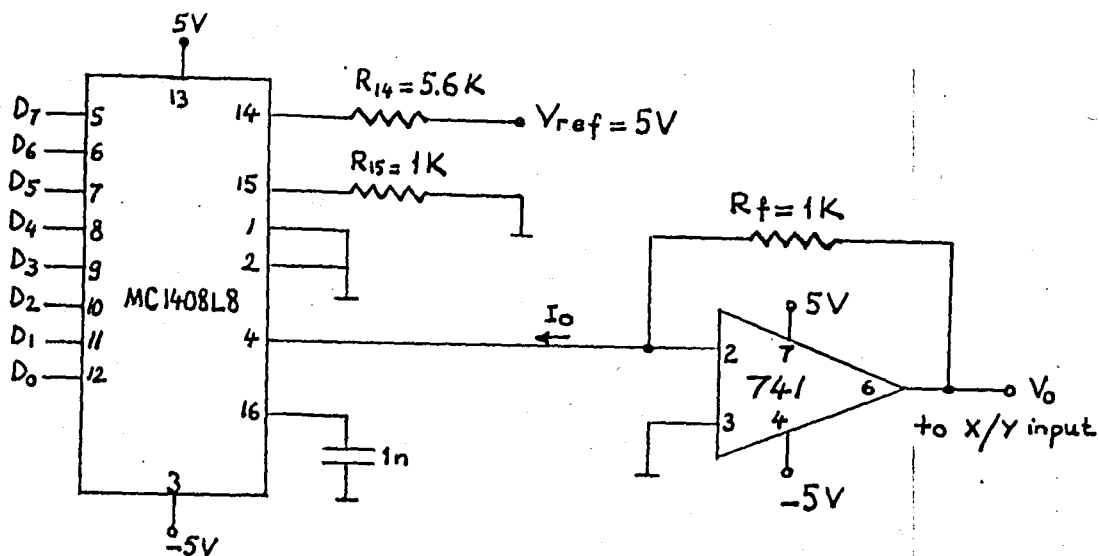


Fig. 4-5 Digital to Analog Converter Circuit.

In order to calculate the component values the following equations should be used:

The output current of the MC 1408L8 is:

$$I_o = K \cdot \left[\frac{D_7}{2} + \frac{D_6}{4} + \frac{D_5}{8} + \frac{D_4}{16} + \frac{D_3}{32} + \frac{D_2}{64} + \frac{D_1}{128} + \frac{D_0}{256} \right]$$

where
$$K = \frac{V_{ref}}{R_{14}}$$

When all the bits of the word are logic ones:

$$I_o = K \frac{255}{256}$$

The output voltage of the op-amp is:

$$V_o = I_o \cdot R_f$$

If R_f is taken as 1 K, then

$$I_o = \frac{V_o}{R_L} = \frac{1}{1 \cdot 10^3} = 1 \text{ mA.}$$

$$K = \frac{256}{255} \cdot 1 \cdot 10^{-3} = 1.0039 \text{ mA.}$$

$$R_{14} = \frac{V_{ref}}{K} = \frac{5}{1.0039} = 4.98 \text{ K.}$$

A 5.6 K resistor is used as R_{14} to get a lower voltage than 1 volt at the output.

$$V_o = \frac{V_{ref}}{R_{14}} \cdot R_f \cdot \frac{255}{256} = \frac{5}{5.6} \cdot 1 \cdot \frac{255}{256} = 0.89 \text{ V.}$$

The pen drop/raise circuit is shown in Fig. 4-6.

The CA2 pin of the MC 6821 PIA is connected to the base of the transistor Q_1 . Q_1 and Q_2 are used as an electronic switch. The microcomputer can drop or raise the pen of the XY plotter. At the beginning the pen is lifted up. When the pen holder goes to the appropriate X and Y coordinates a high level voltage is applied to the base of Q_1 by CA2 and the pen drops down. CA2 goes low and the pen is raised when the drawing is finished.

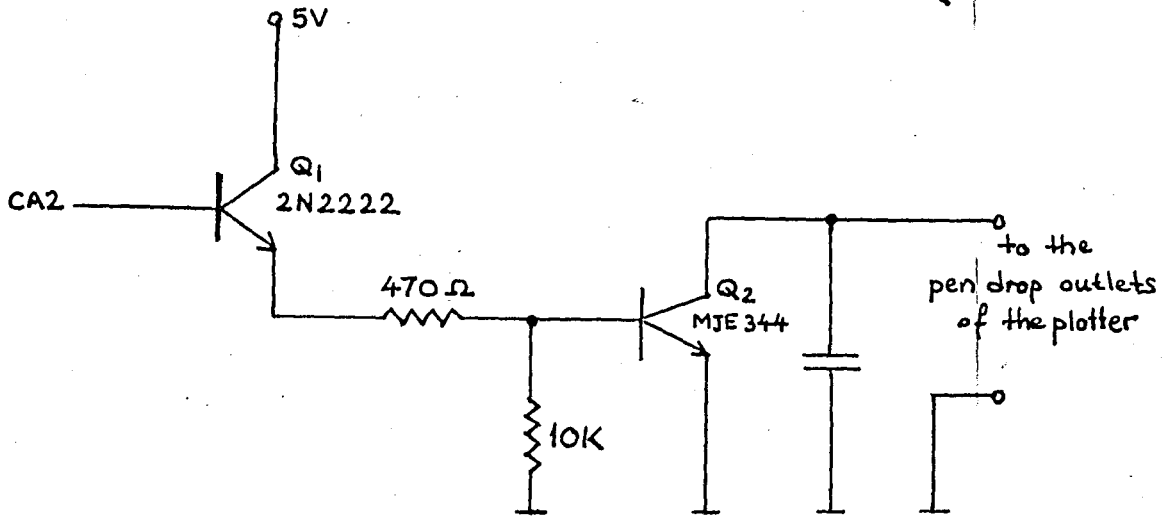


Fig.4-6 Pen Drop/Raise Circuit.

The alarm circuits are used to generate the appropriate signals when the position of the marker pen is outside of the map. One of the two alarm signals indicates that the position of the target is outside and the other warns us for our ship. These signals can be used for automatic changing of the paper in further developments. Fig.4-7 shows the alarm circuit..

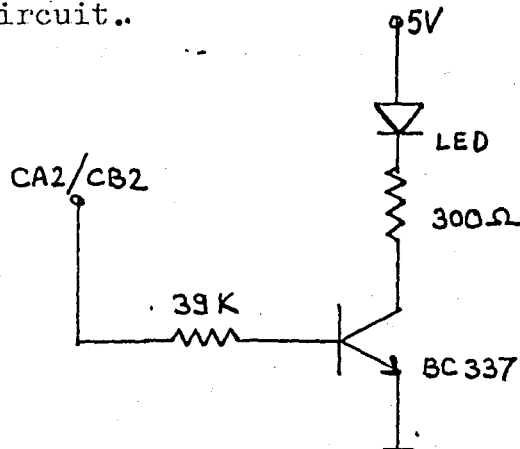


Fig.4-7 Alarm Circuit.

Timing Circuit

The positions of the ship are marked at regular intervals. Timing circuit generates an interrupt signal to the CPU when the plotting time is reached. The processor executes the interrupt service routine and the position of the ship is marked on the paper.

In Fig.4-8 the 555 timer is connected for astable operation.

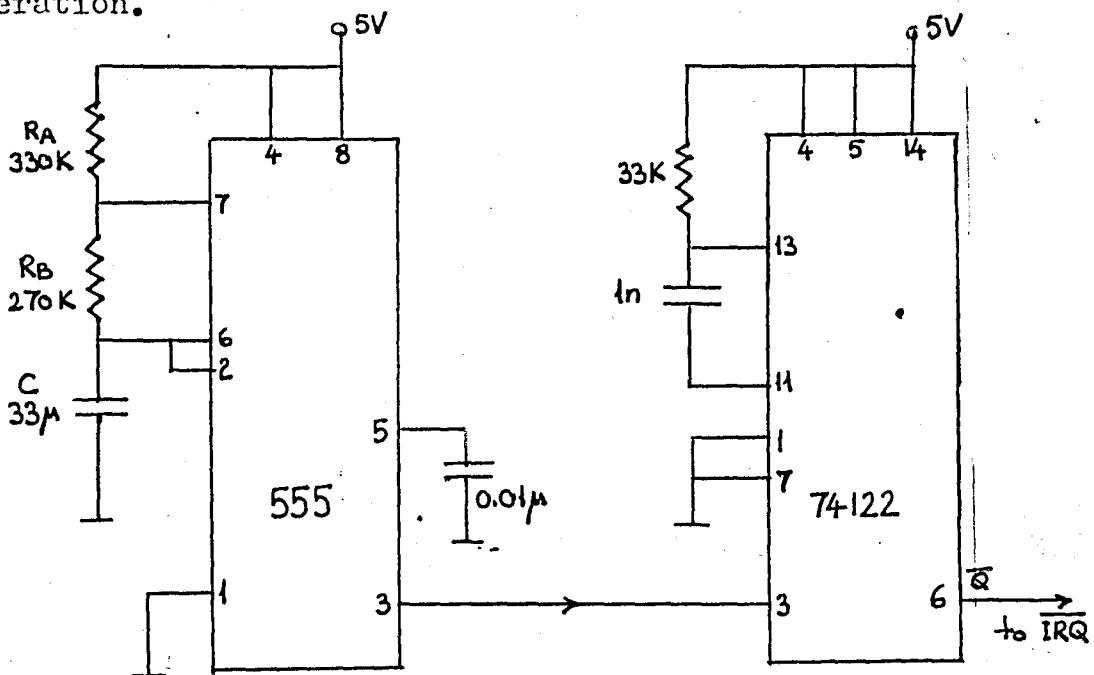


Fig.4-8 Timing Circuit

The period of the output signal is

$$T = 0.693 (R_A + 2R_B) \cdot C$$

$$T = 0.693 (330 \cdot 10^3 + 2 \cdot 270 \cdot 10^3) \cdot 33 \cdot 10^{-6}$$

$$T = 19.9 \text{ s.}$$

The output of the 555 is fed to the 74122 Monostable Multivibrator to get the $10 \mu\text{s}$. length interrupt request pulses. The waveforms are shown in Fig.4-9

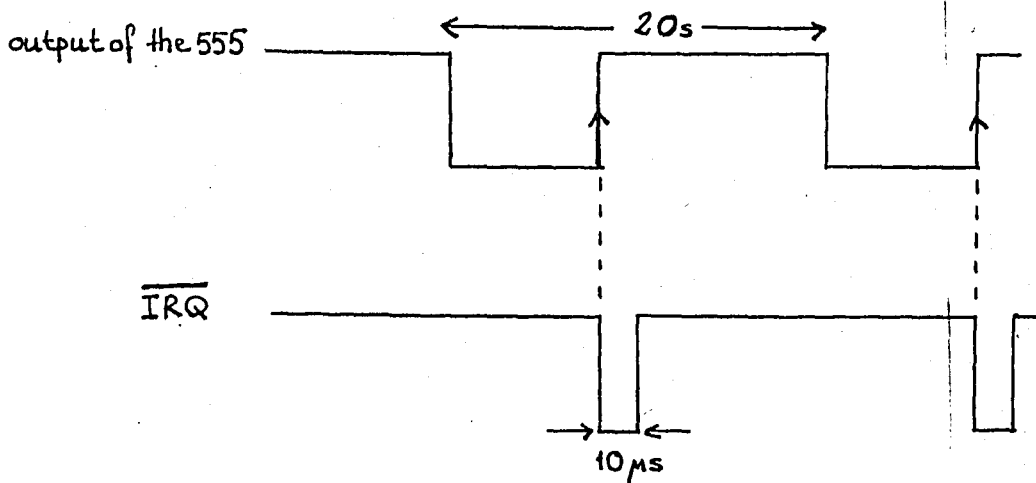


Fig.4-9 Waveforms

The time intervals between the position markings may be changed by using different resistors and capacitors in the timing circuit. Variable plotting frequencies are useful to mark the positions of the ships having rapid variations in their courses. If a 500 K potentiometer is used instead of 270 K resistor the plotting interval may be adjusted in the range of 7-30 s. The scale number should be calculated after the adjustment of the time interval.

CHAPTER 5

SOFTWARE DEVELOPMENT

The program of the system includes two parts: Main Program and Interrupt Service Routine. In this chapter the two parts of the program will be described completely.

Main Program

The processor will begin to execute the Main Program when the system is turned on. The program initializes the PIA's as shown in Fig.5-1. The situations of the PIA's are as follows:

DF00	P1ORA	X output to the plotter.
DF01	P1CRA	Pen_drop signal (CA2)
DF02	P1ORB	Y output to the plotter.
DF03	P1CRB	Own ship alarm indicator (CB2)
DF04	P2ORA	Own course input
DF05	P2CRA	Target alarm indicator (CA2)
DF06	P2ORB	Own speed input
DF07	P2CRB	Spare alarm (CB2)
DF08	P3ORA	Reset x input

DFO9	P3CRA	
DFOA	P3ORB	Reset y input
DFOB	P3CRB	
DFOC	P4ORA	Target Bearing input
DFOD	P4CRA	
DFOE	P4ORB	Target Distance input
DFOF	P4CRB	

After initialization, data present on peripheral lines programmed as inputs are transferred to the memory. These are, own course, own speed, target bearing, target distance and reset or reference position inputs. One another input to the system is the Scale. It would be desirable to have selectable scales. The scale to be used is determined by a combination of the size of the plotting surface, 10" by 10", and the fact that an 8-bit D to A converter is selected. Using a scale of 500 yards per inch and a sample time of 20 seconds results in a scale factor 0.56666 which, when multiplied by the speed of the ship, gives the number of bits to output for ship travel in a 20 second period of time. The following equation shows this calculation:

$$S \frac{\text{Miles}}{\text{Hour}} \times 2000 \frac{\text{Yards}}{\text{Mile}} \times \frac{1}{3600} \frac{\text{Hours}}{\text{Second}} \times \frac{1}{500} \frac{\text{Inches}}{\text{Yard}}$$

$$\times 20 \text{ Seconds} \times 25.5 \frac{\text{bits}}{\text{inch}} = 0.56666 \times S \text{ Bits}$$

where S is the speed of the ship. Because an eight-bit converter is used, 1 inch corresponds $\frac{255}{10} = 25.5$ bits on the plotting surface.

The calculations for the target needs another scale. Target Scale is calculated by a multiplication subroutine as follows:

$$100 \times D \text{ yards} \times \frac{1}{500} \frac{\text{Inches}}{\text{Yard}} \times 25.5 \frac{\text{bits}}{\text{inches}} = D \times 5.1 \text{ Bits}$$

In this example 1 inch corresponds 500 yards, but in different operations the various scales can be used i.e. for a man-overboard operation 1 inch \Rightarrow 200 yards, for a gunnery operation 1 inch \Rightarrow 694 yards.

The own course and target bearing are in Gray code and must be converted to binary. The resulted binary number is between 0-255. In order to find the real course between 0-359, the binary number is multiplied by $\frac{360}{256} = 1.4265$. The program clears the target flag (TARGET) and starts to the calculations for own ship. The first duty is to find the quadrant of the course. After determining the quadrant, the course will be an angle between 0-90°.

Since the object of these calculations is taking polar coordinates in and outputting cartesian coordinates,

it is necessary to do some trigonometric operations. Because the inputs are converted in whole degrees it is possible to write a program which makes all necessary operations with only a table of sines from 0° to 90° . The use of a look-up table increases the speed and provides reduction in instructions.

After reducing the angle to less than 90° the table is accessed to get the sine, then the angle is subtracted from 90° and the sine table again accessed, but this time to get the cosine of the original angle.

Because a 16 bit binary number is used to represent the sine accurately, it is necessary to utilize a 16 bit by 16 bit multiplication routine. The scale factor, as a 16 bit binary number, is first multiplied by sine. The scaled sine is multiplied by the speed in order to get the y component of the departure of own ship. The same algorithm is repeated for x axis. Therefore, x and y components of the movement in unit time, is ready for further calculations. The flow charts of the algorithms above as shown in Fig.5-12.

The program returns back to "AGAIN" point and repeats same algorithm by loading the target bearing and distance to the working registers. During this part of the program Target

Scale is used as the scale factor in calculations. At the end of this routine the x and y components of the distance between own ship and the target are determined and stored.

The algorithm of the main program is completed by clearing the interrupt mask bit. The processor returns to the "BEGIN" point and reads all the inputs again. Since the execution time of the main program is very short, the variations at the all inputs are detected immediately. The calculated values are corrected with the new data at the inputs. When there is no interrupt request signal the system executes the main program repeatedly. The x and the y components of the departure of both own ship and the target are renewed by the latest input values.

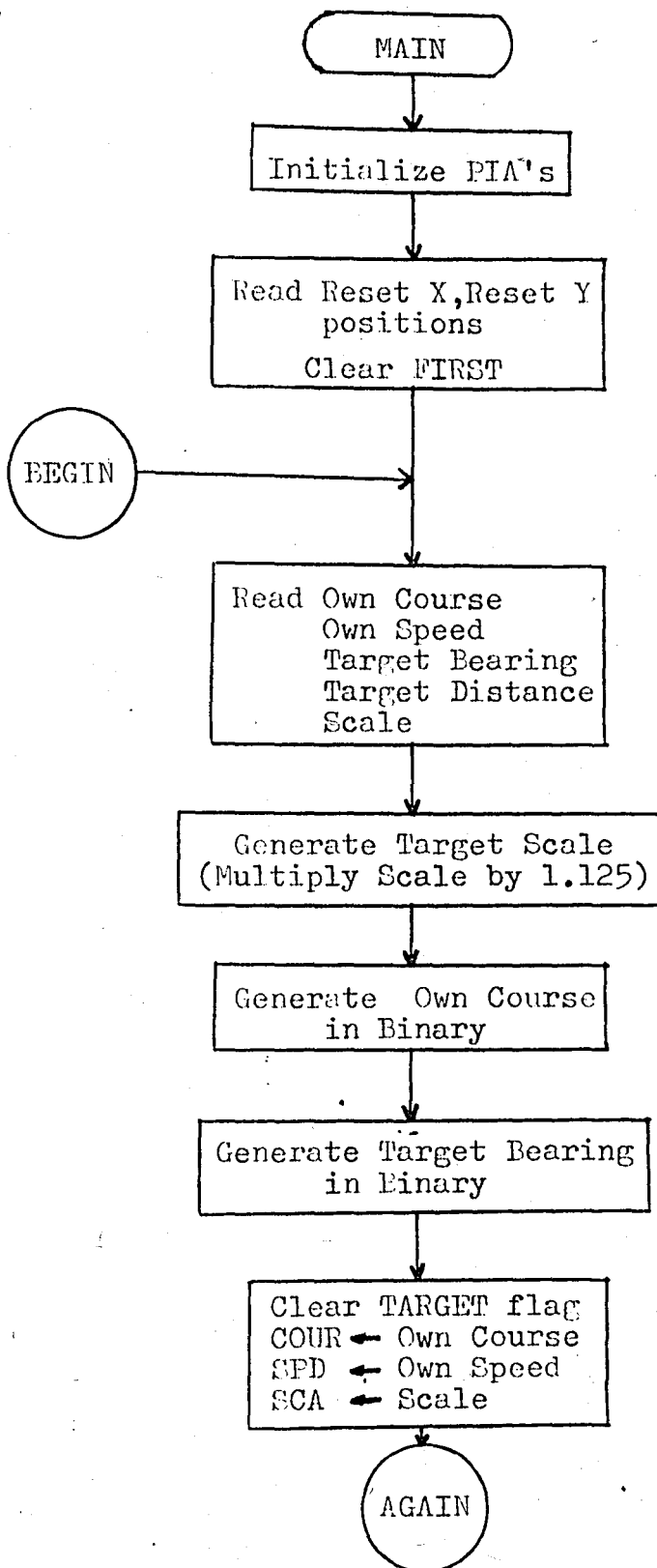


Fig. 5-1

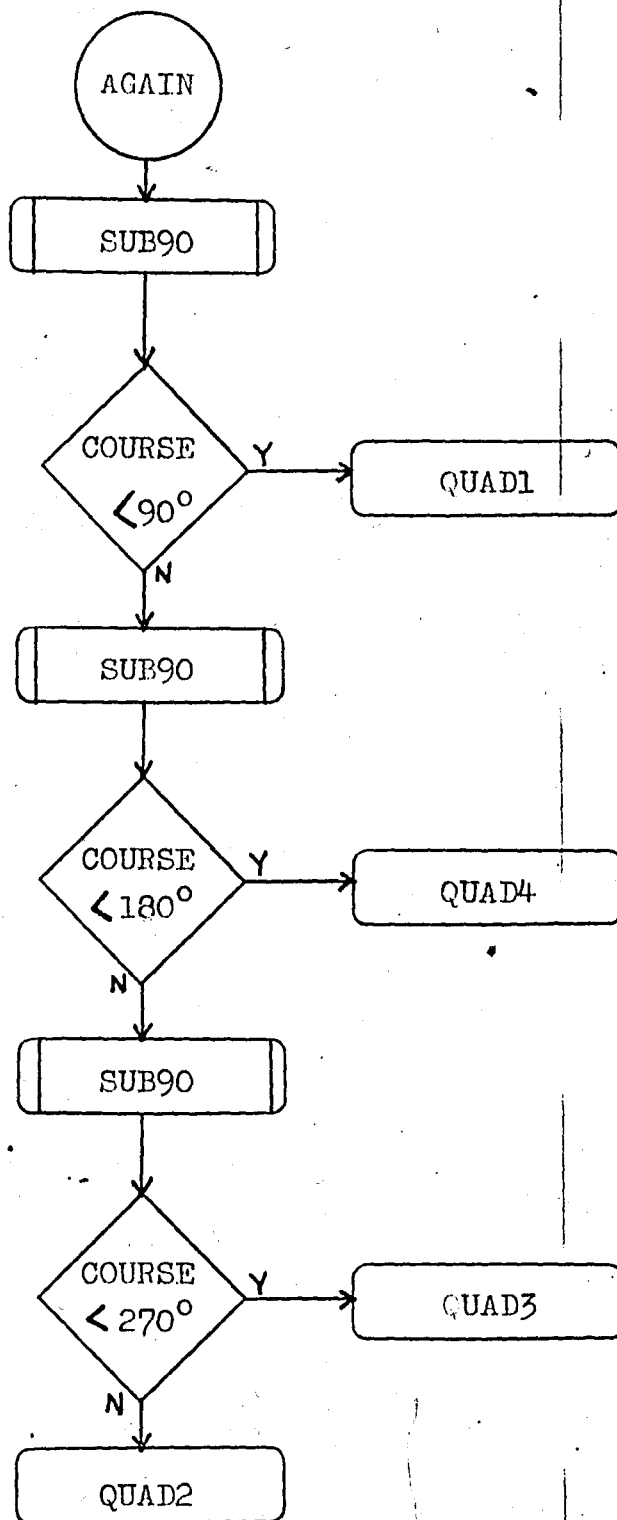
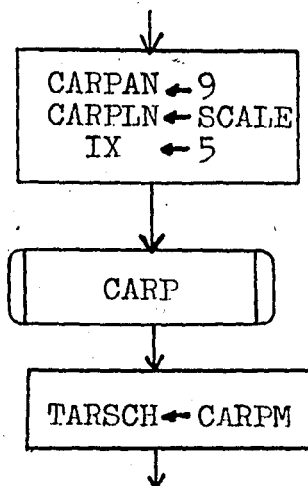


Fig. 5-2

Generation of target scale:



Generation of own course/target bearing:

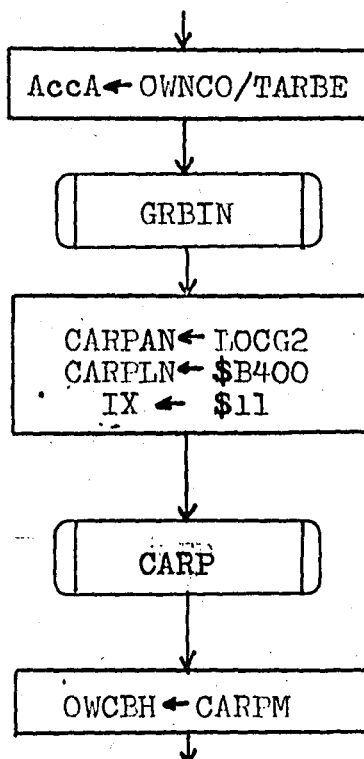


Fig. 5-3

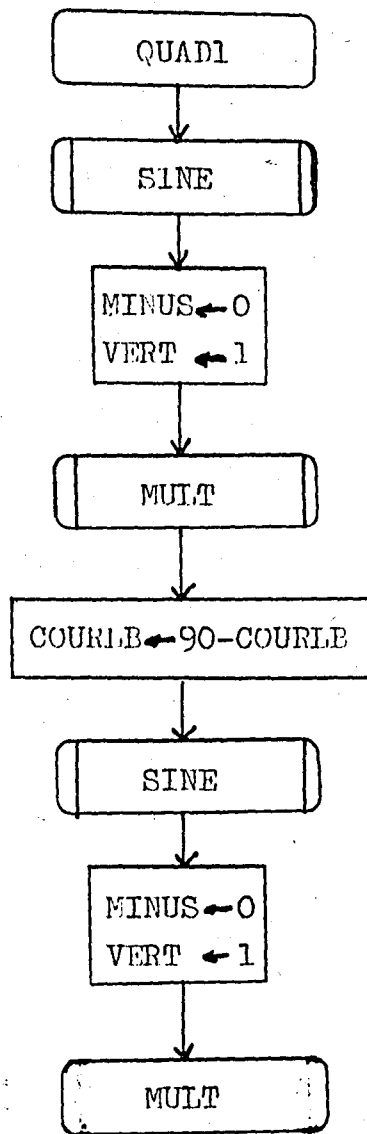


Fig. 5-4

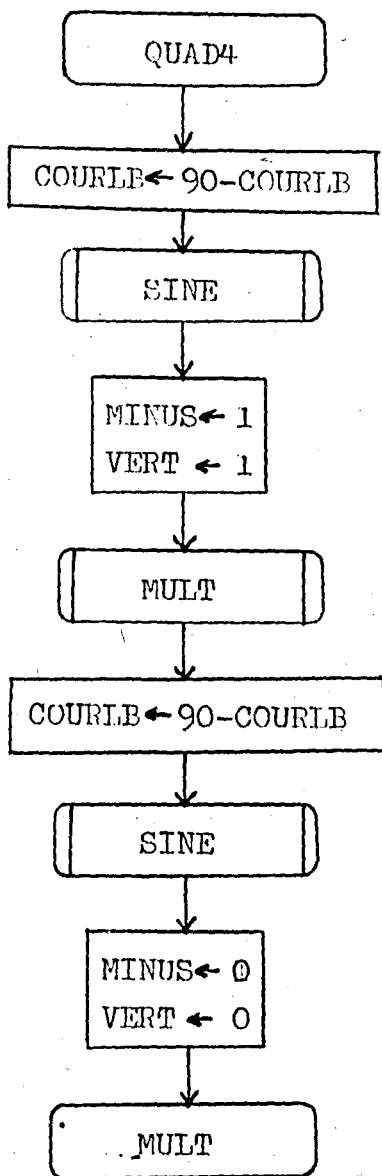


Fig. 5-5

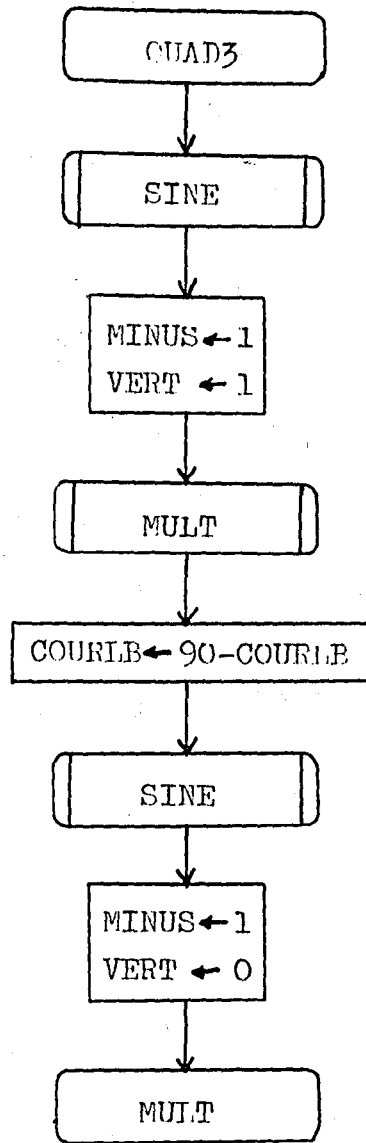


Fig. 5-6

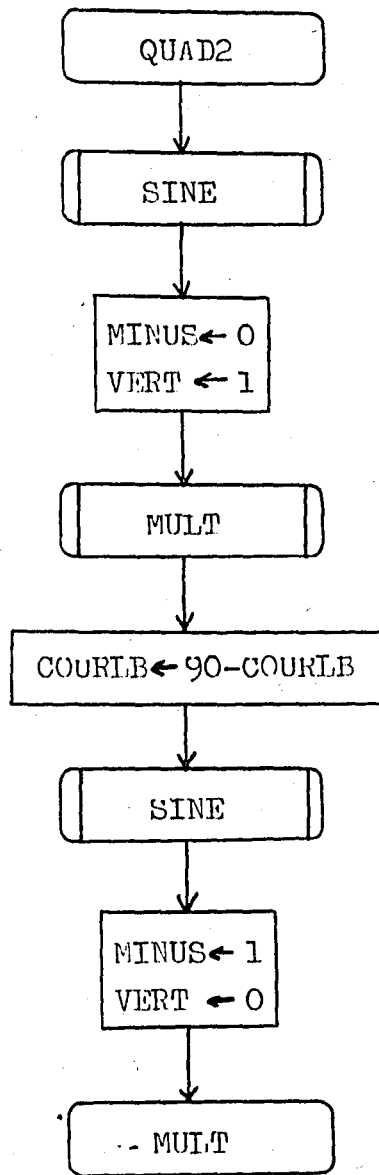


Fig. 5-7

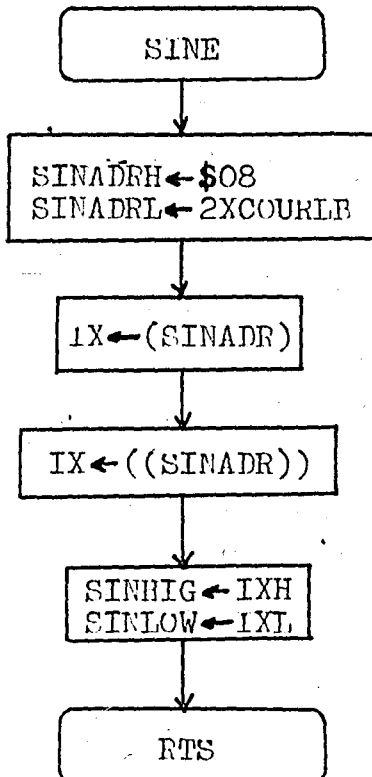
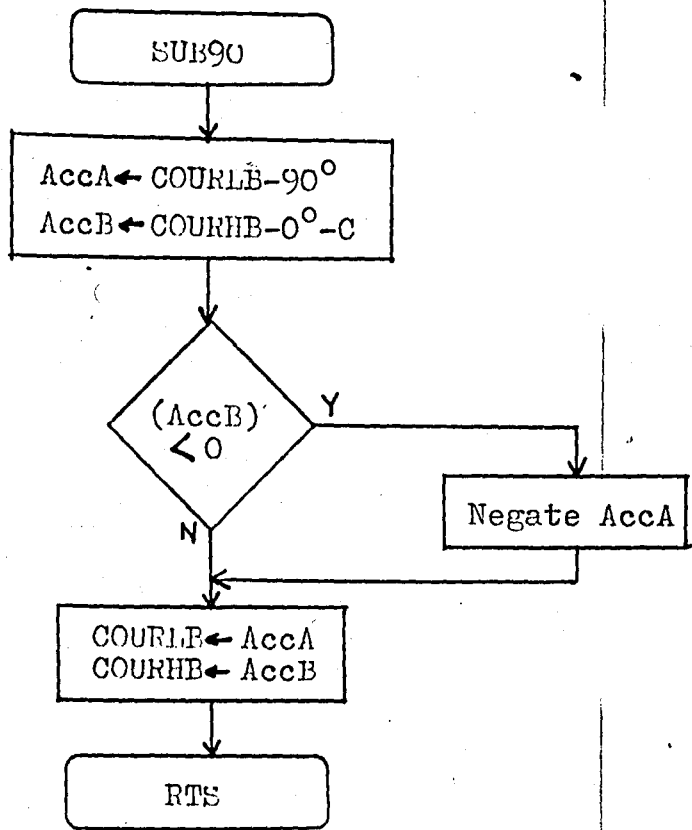


Fig. 5-8

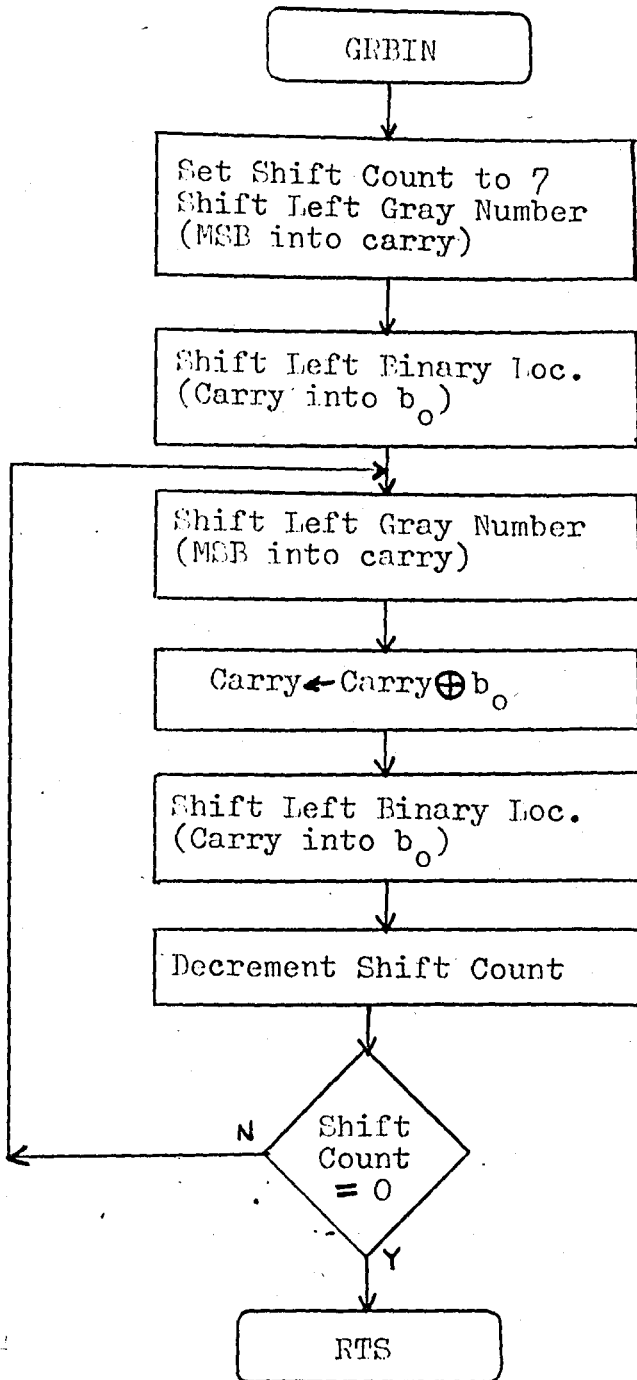


Fig. 5-9

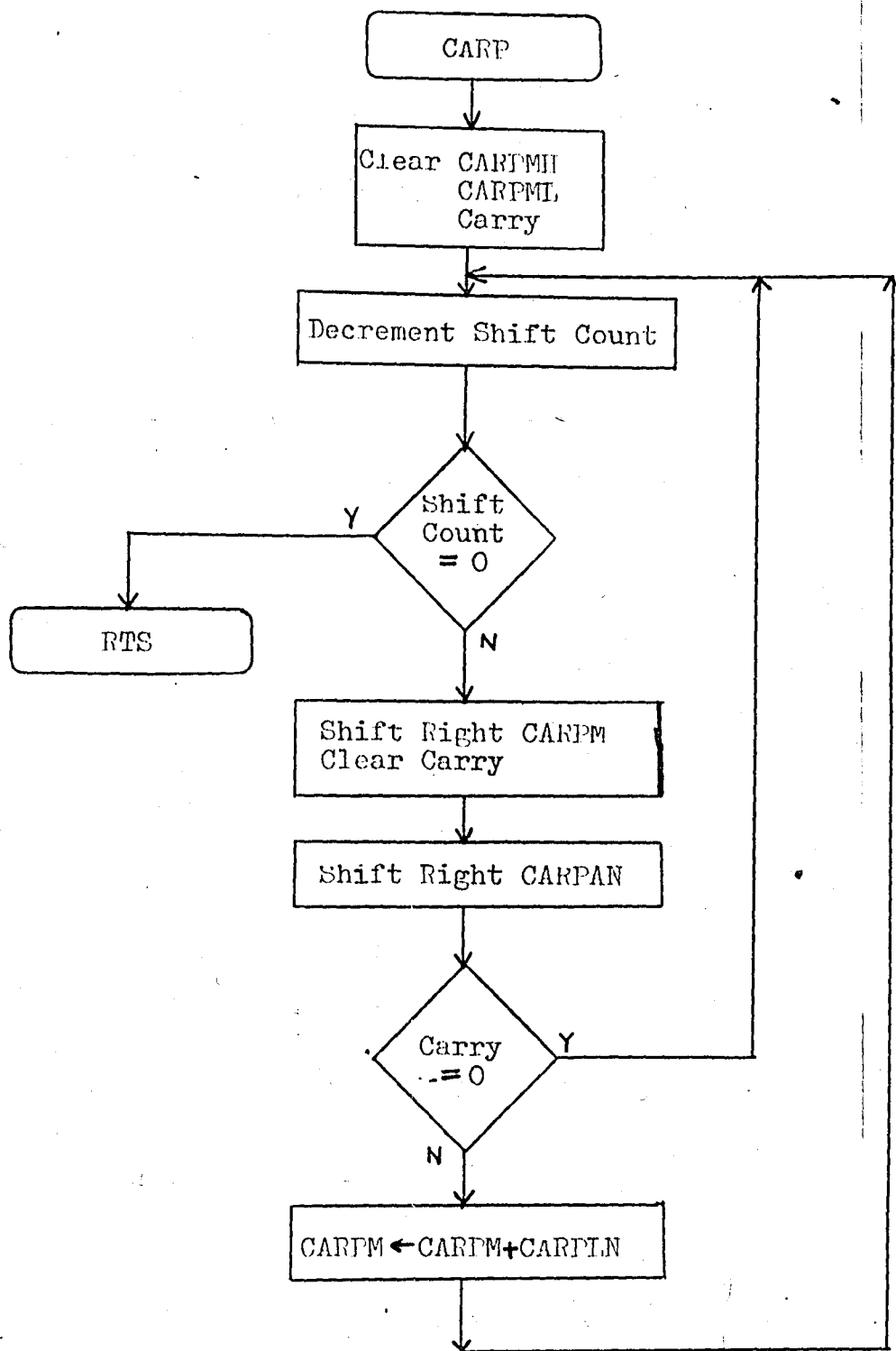


Fig. 5-10

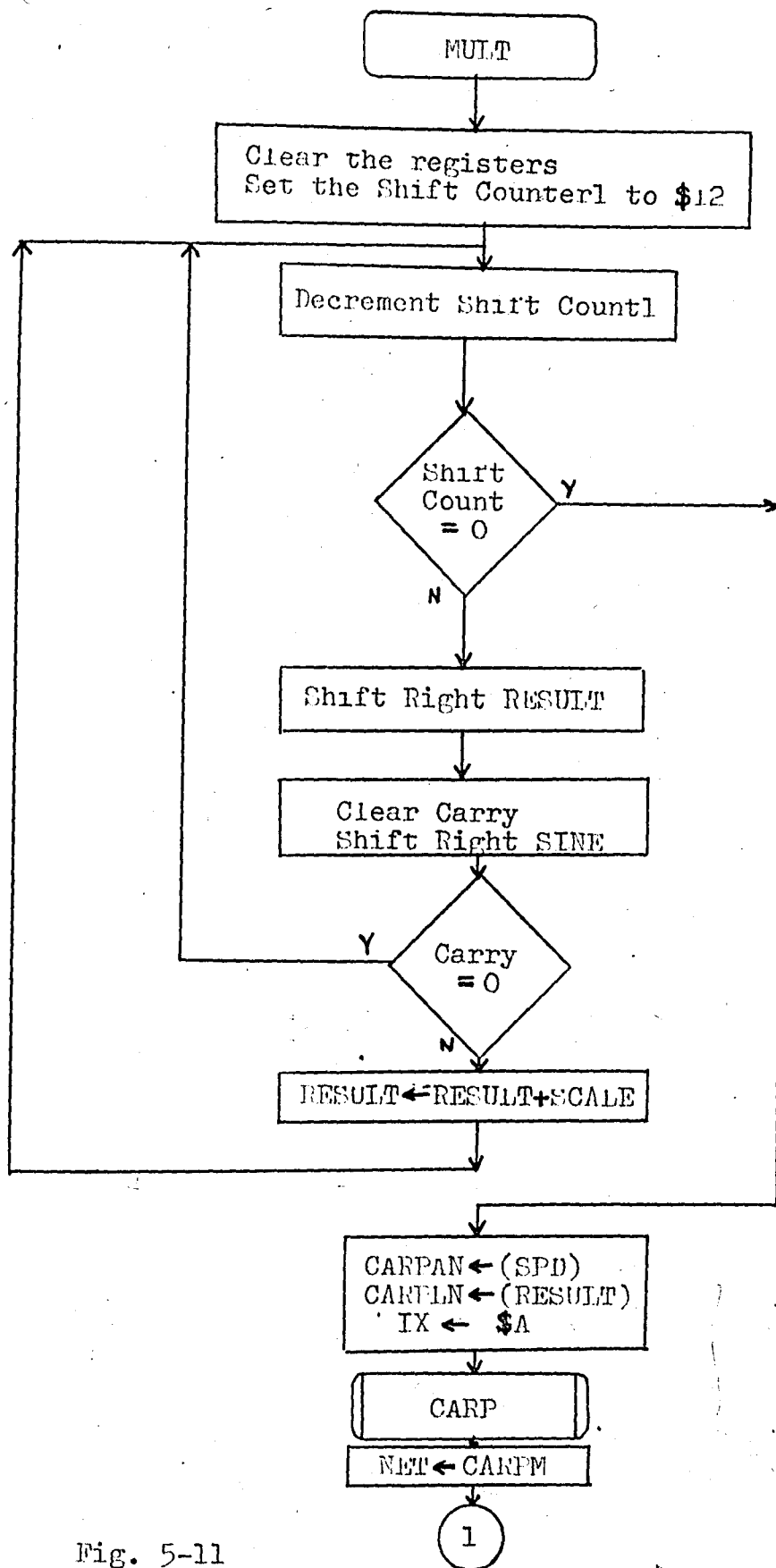


Fig. 5-11

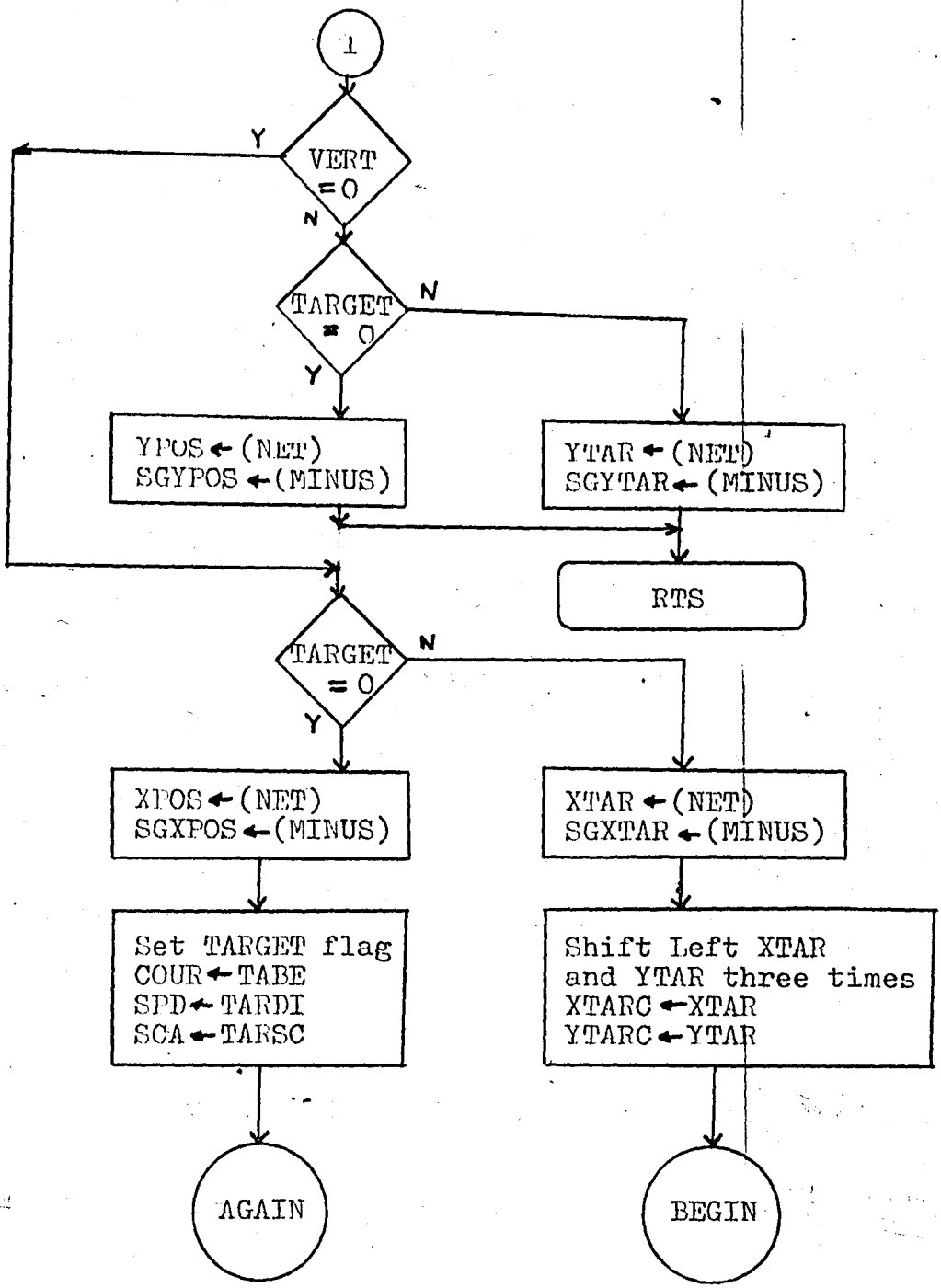


Fig. 5-12

Interrupt Service Routine

When the interrupt request signal is detected, the processor jumps to the service routine. The binary coded information about the course and the speed of our ship are converted to Binary Coded Decimal (BCD) digits. The data about own ship are transmitted to the teletype in the following form:

OWN COURSE : 356

OWN SPEED : 015

where the unit of the course is degree and the unit of the speed is mile per hour (knot).

The coordinates of the position of our ship are determined by the addition of the departure component to the reference coordinate if the movement is in positive direction, otherwise the departure is subtracted from the reference coordinate. The position of the ship is examined in order to learn whether it is inside of the plotter boundaries. If the position is outside of the plotter, CB2 output of the PIA 1 is set to inform the alarm circuit and the processor returns from interrupt service routine. If the position is plottable, a square figure is drawn by PLOTME subroutine.

The reference for the target position is the position of our ship. The coordinates of the target position are calculated with the x and y components of the distance. These components are added to the coordinates of our ship or subtracted from them. If the distance between the target and our ship is zero, the system thinks that there is not any target ship in the area and returns from the interrupt routine..

After calculating the target position, it is checked if the position is inside of the plotter surface. If it is outside, CA2 pin of the PIA 2 is set in order to transmit an alarm signal.

During the first pass through this routine, the course and the speed of the target can not be determined since there is not any knowledge about the previous position. In order to understand whether this is the first pass or not, the "FIRST" flag is checked. The situation of this flag is complemented during the first pass. During the second or any other passes, the difference between the present and the former position stored before is calculated. After some trigonometric operations the course and the speed of the target are ready to be printed on the teletype paper in the following form:

TARGET BEARING : 073
 TARGET DISTANCE : 01500
 TARGET COURSE : 187
 TARGET SPEED : 021

where the unit of the distance is yard. This routine is finished by drawing a diamond shape on the position of the target. The interrupt service routine flowcharts are shown in Fig. 5-13 to 5-25.

During the target course and speed calculations, Newton-Raphson method is used for computing the square-roots. The general formula of this method is⁸:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad n=0,1,\dots$$

Let $x = c^{1/2}$, hence $f(x) = x^2 - c = 0$

$$f'(x) = 2x$$

and

$$x_{n+1} = x_n - \frac{x_n^2 - c}{2x_n} = 0.5\left(x_n + \frac{c}{x_n}\right)$$

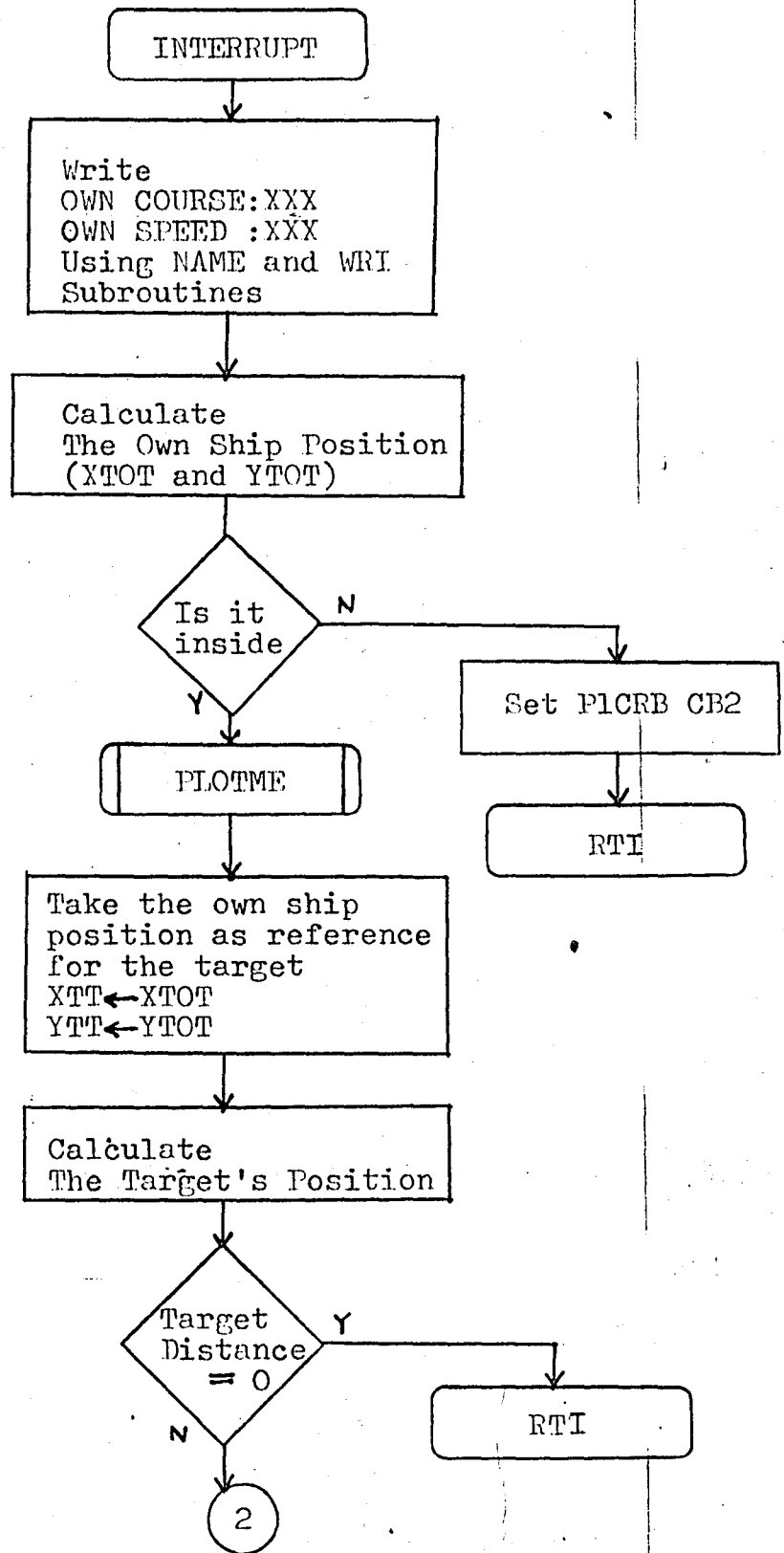


Fig.5-13

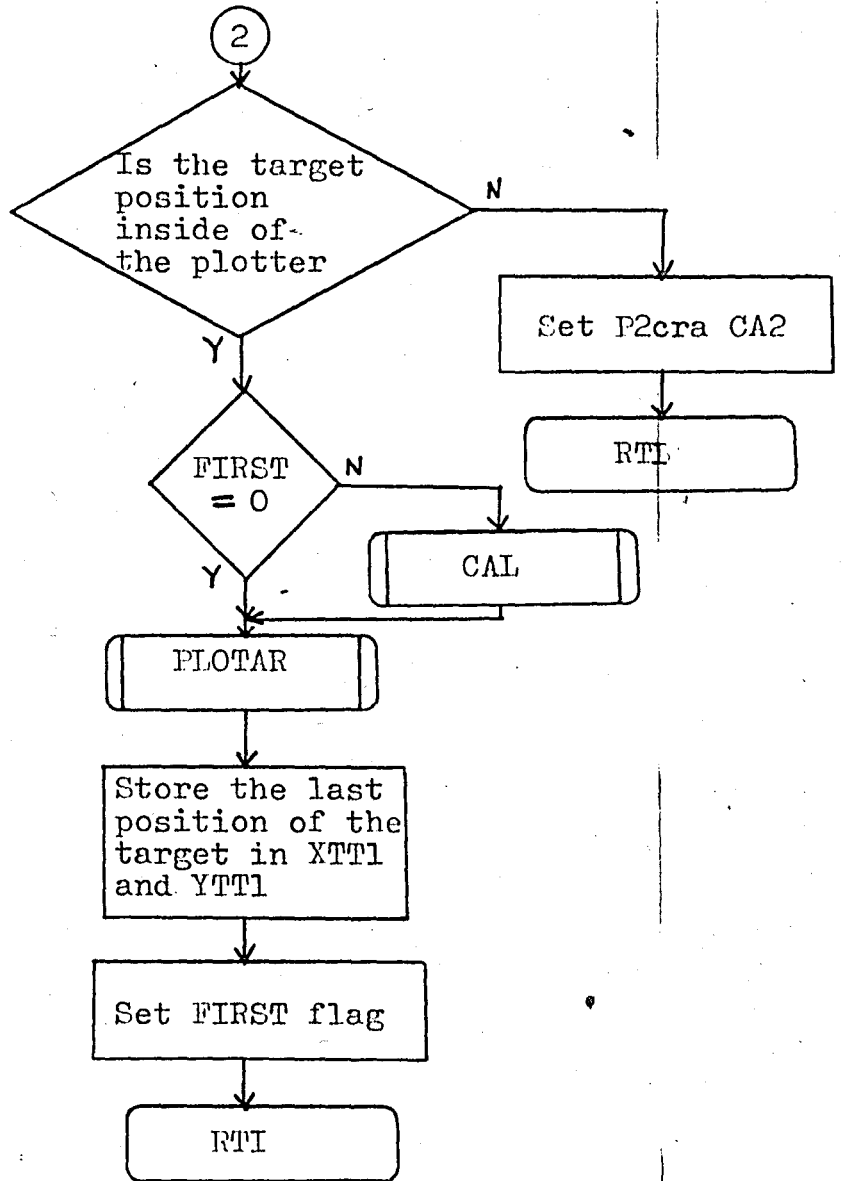


Fig.5-14

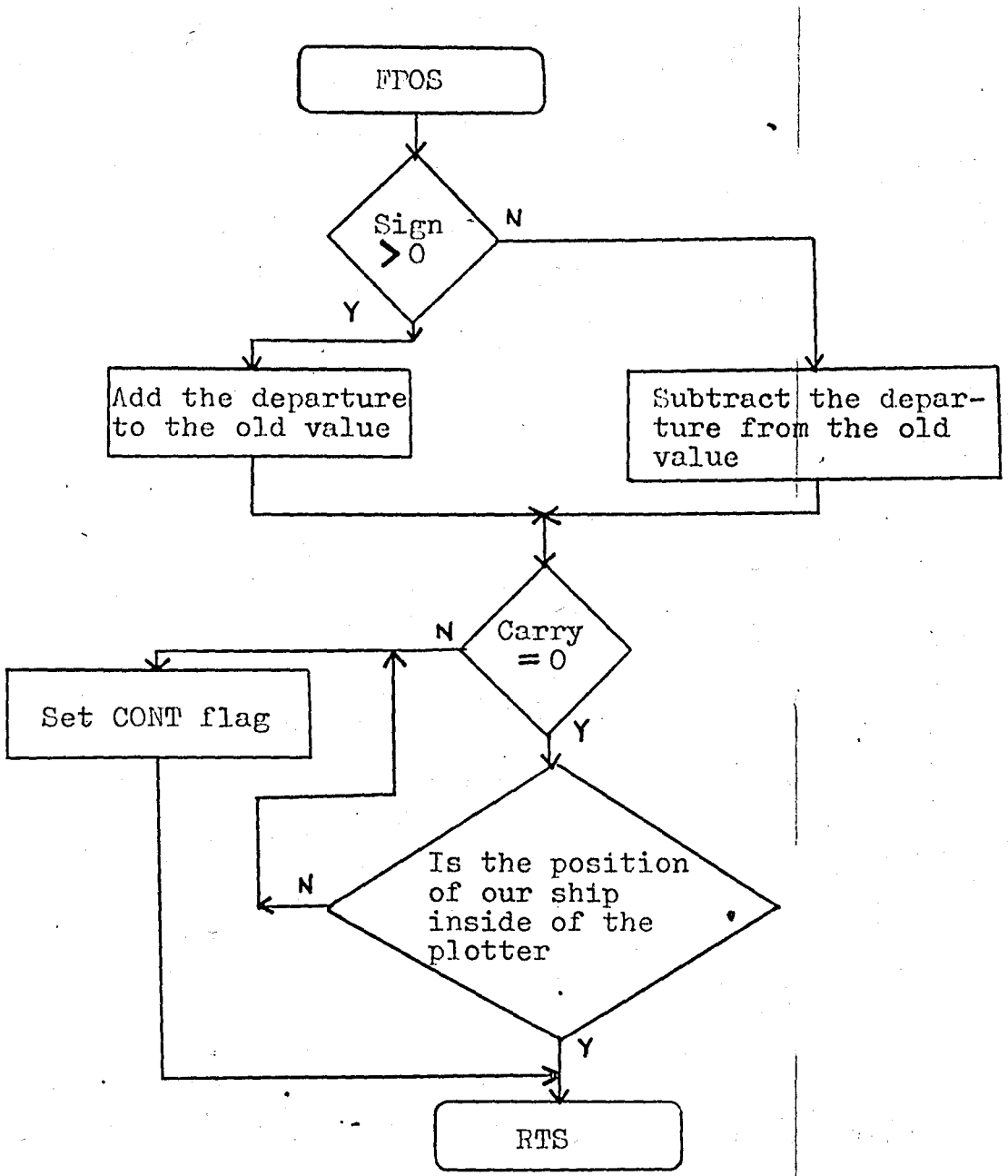


Fig.5-15

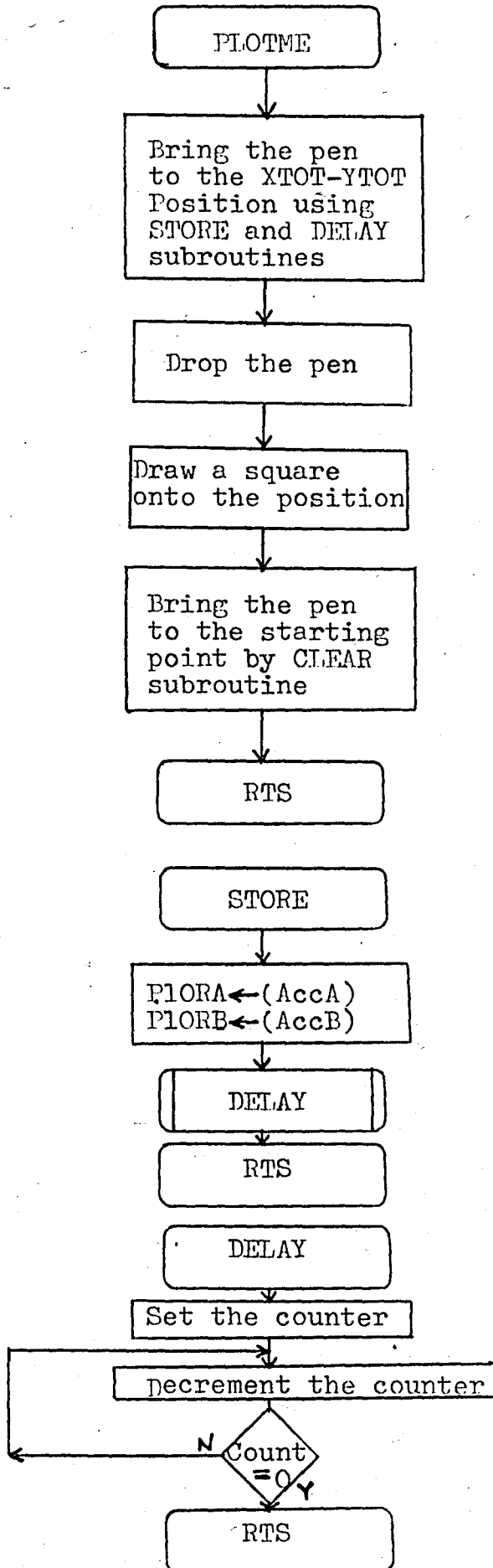


Fig.5-16

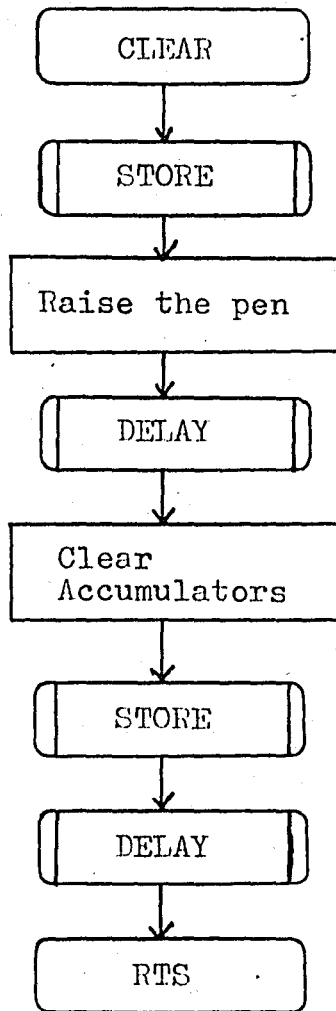


Fig.5-17

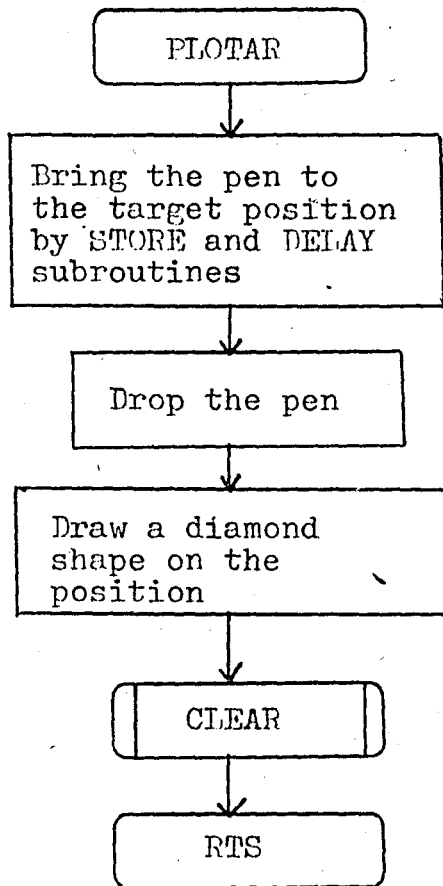


Fig. 5-18

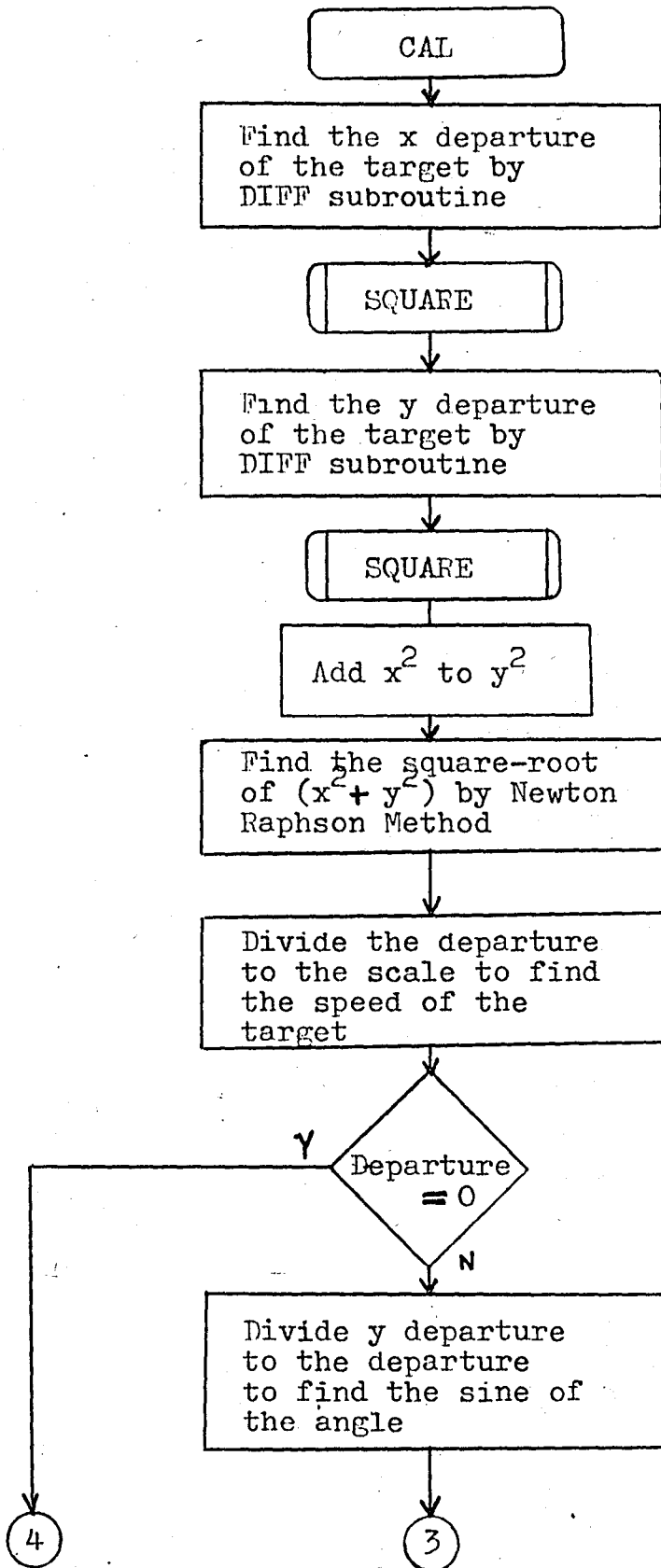


Fig.5-19

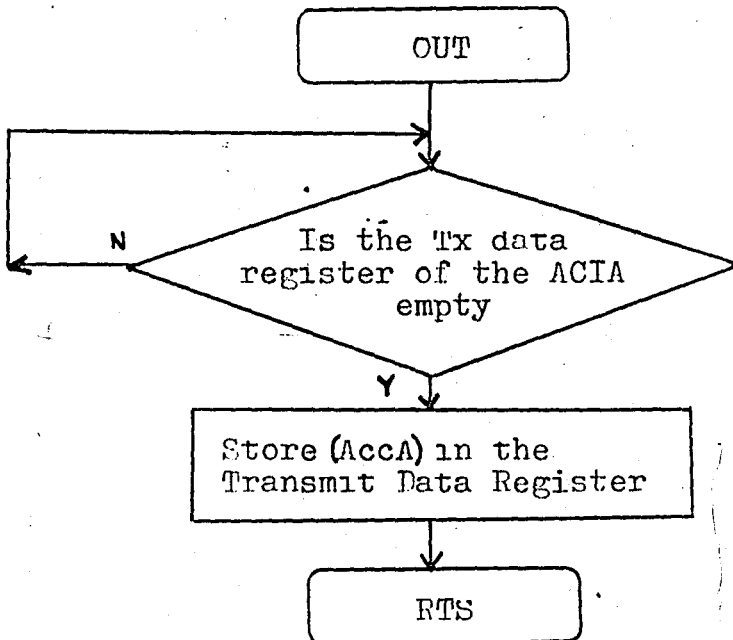
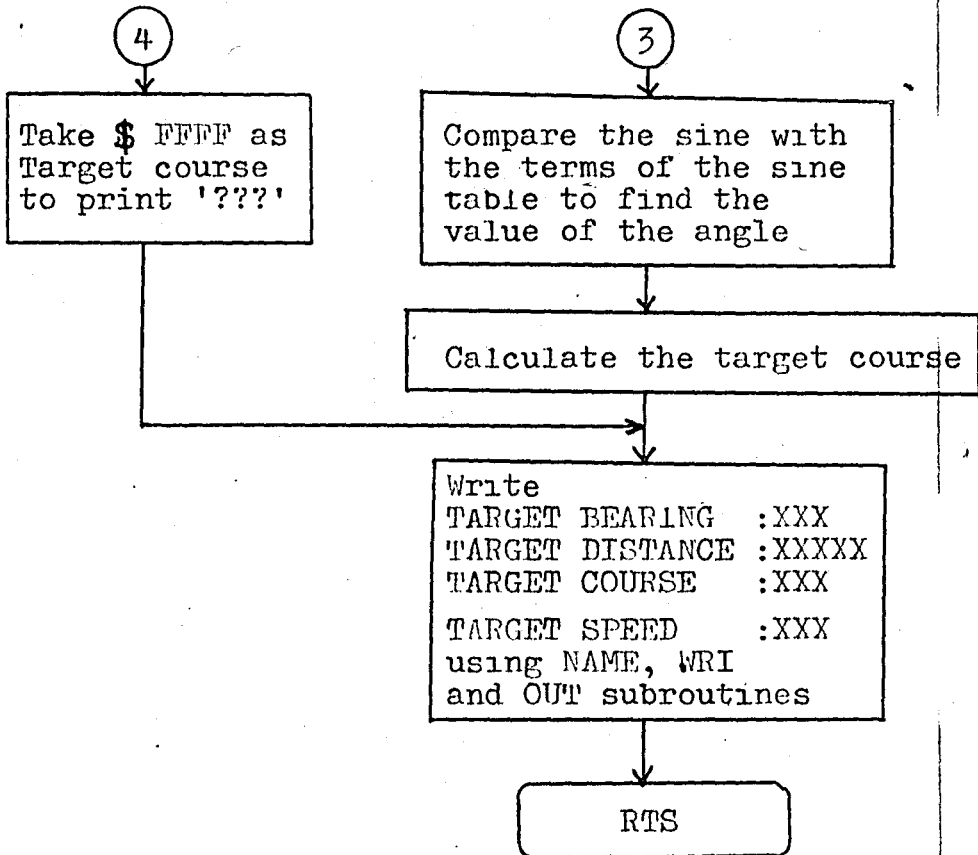


Fig.5-20

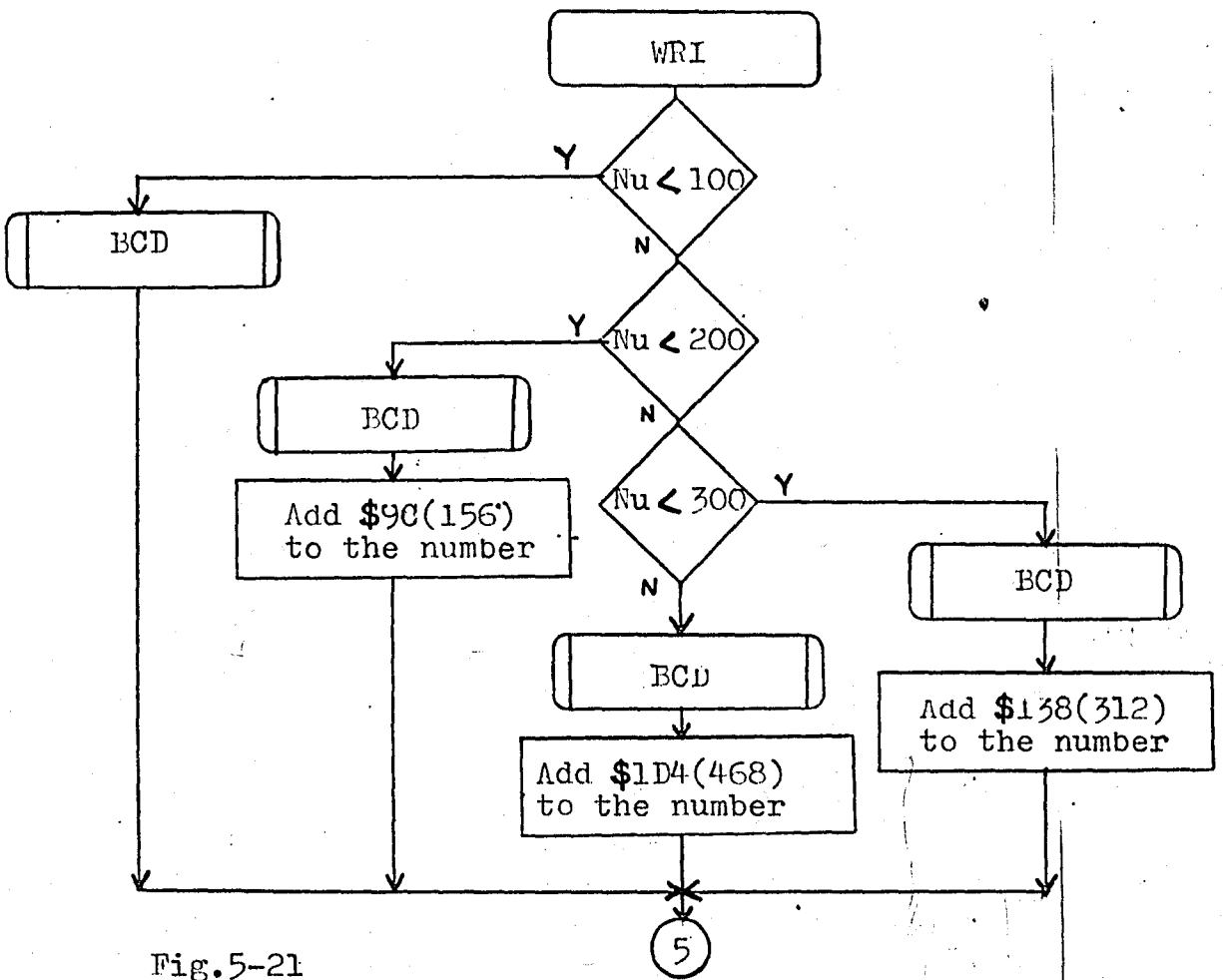
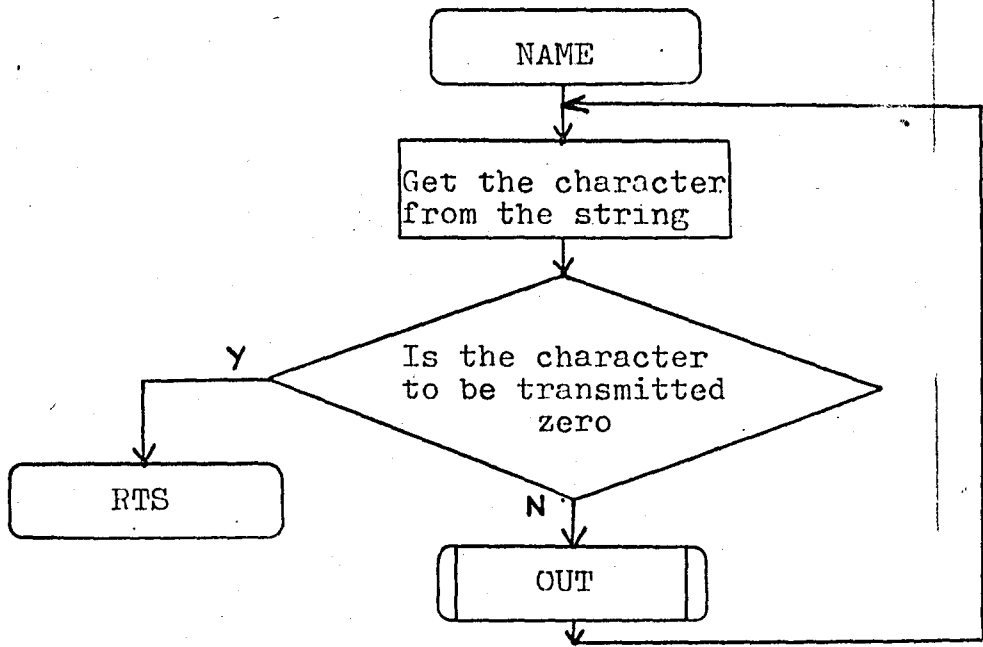


Fig.5-21

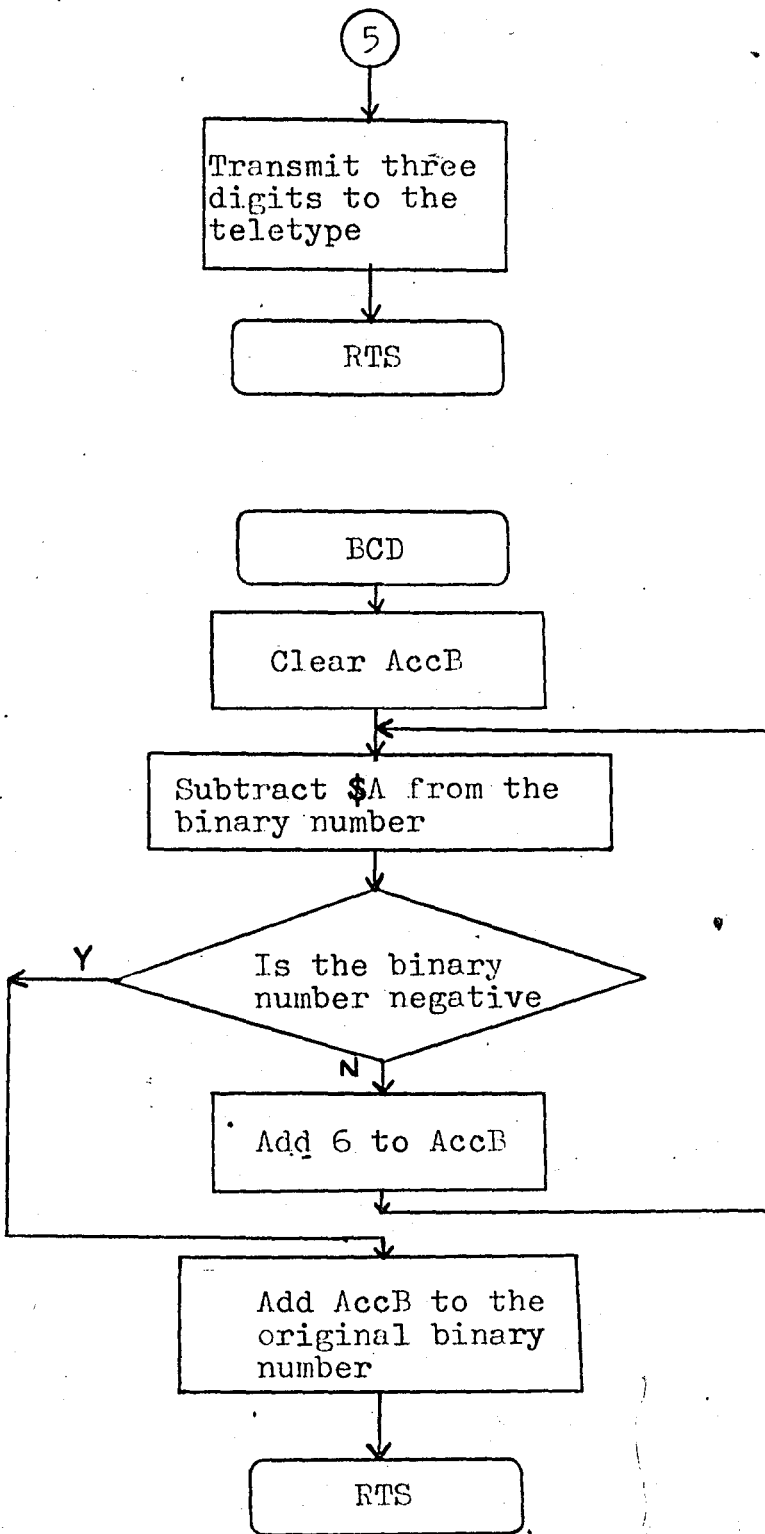


Fig. 5-22

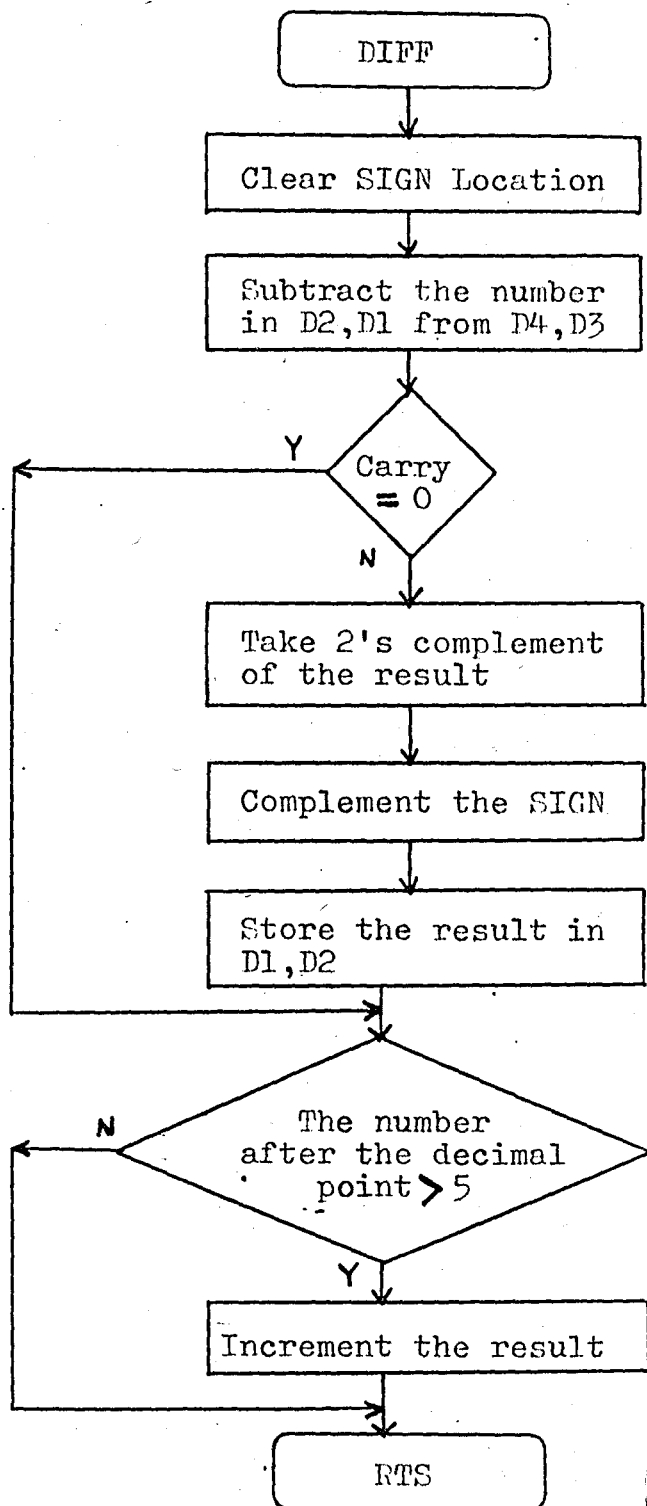


Fig. 5-23

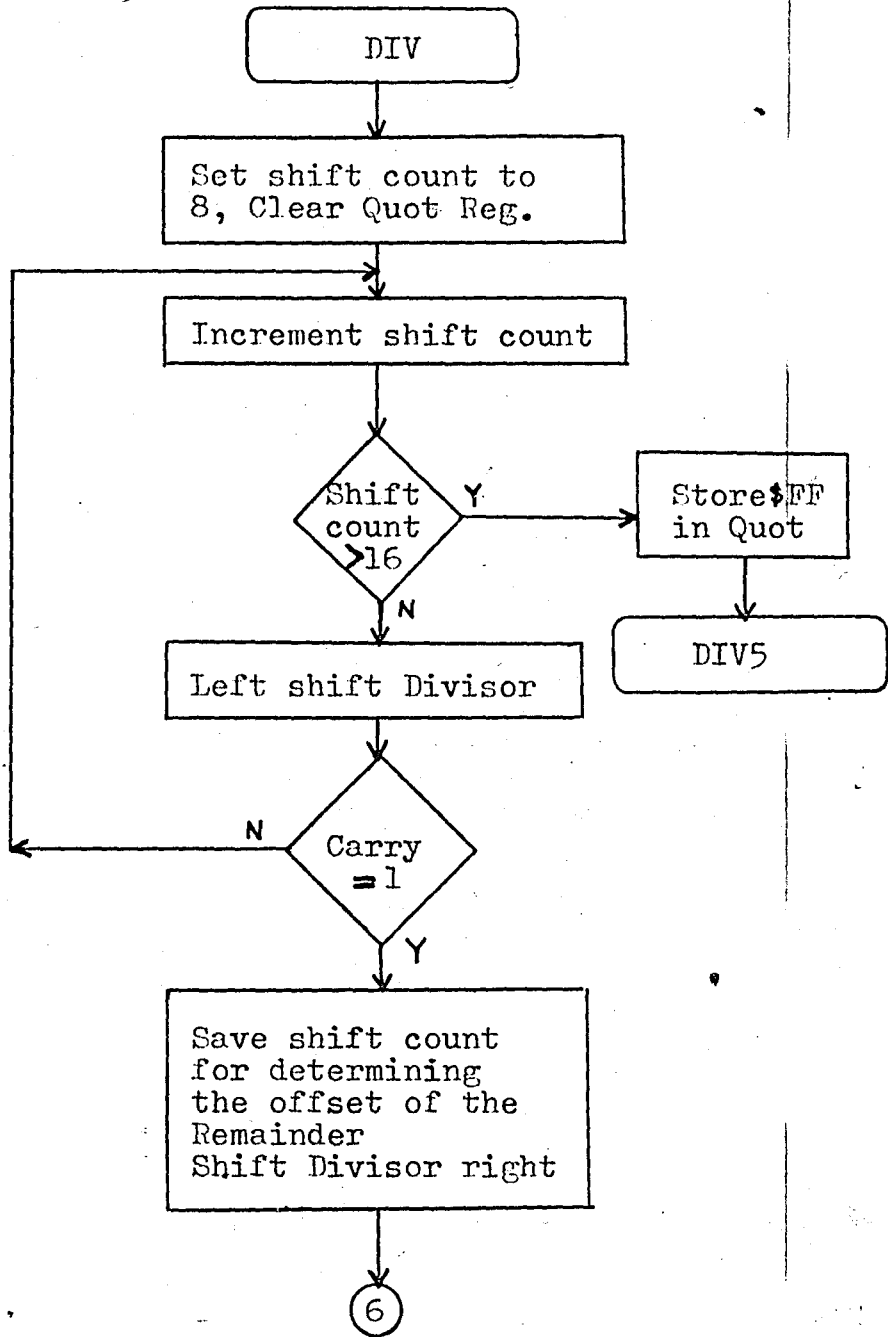


Fig.5-24(a)

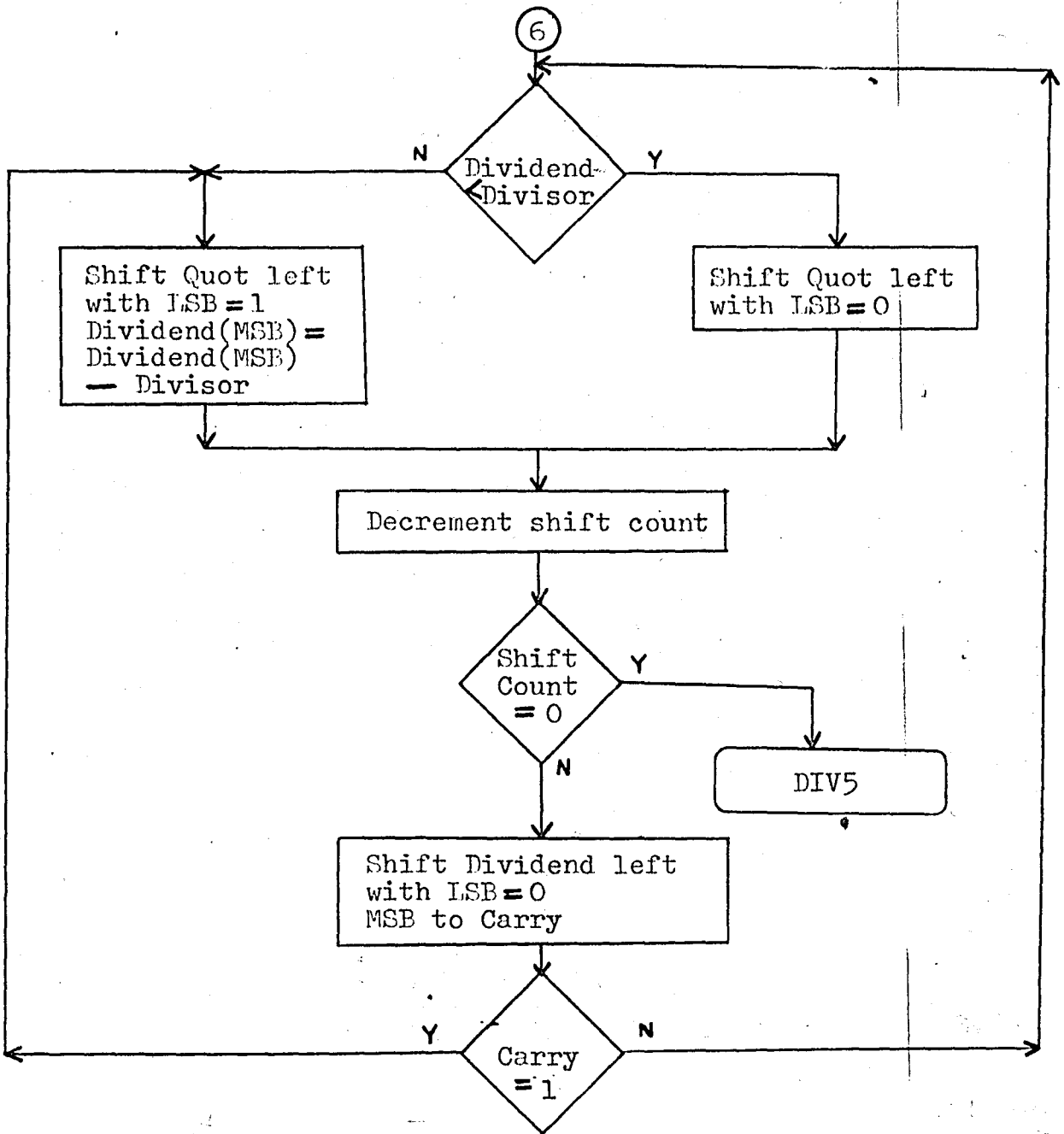


Fig. 5-24(b)

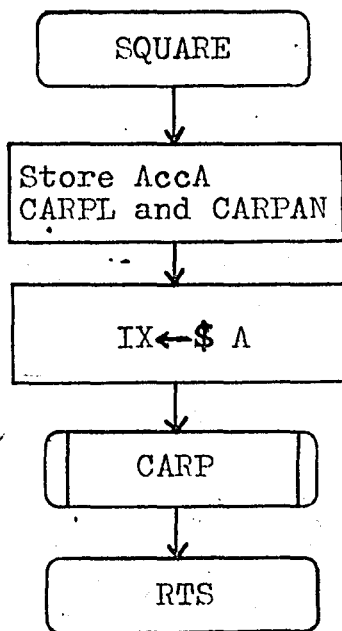
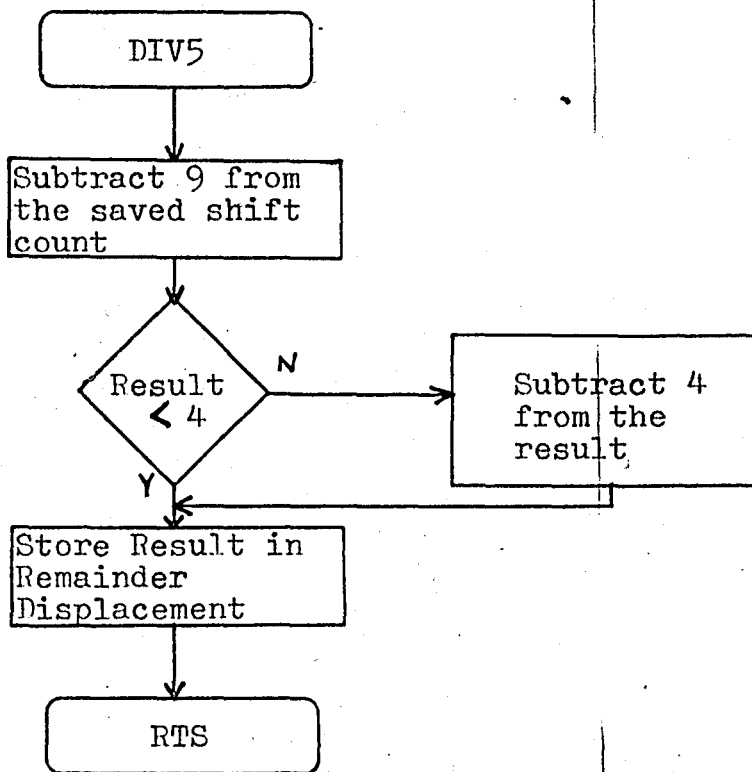


Fig.5-25

CHAPTER 6 DISCUSSION

A sixteen-bit microprocessor will be very useful for this system. There are many subroutines with 16 bit operations in the program. A 16-bit microprocessor will decrease the execution time and provide reduction in instructions.

The use of an eight-bit digital to analog converter results some difficulties. The range of the plotter is 255 bits with this converter. A ship having a departure of 10 bits in a sampling period, reaches to the edge of the paper at the 25th mark. This number may be increased using a sixteen bit D/A converter.

CHAPTER 7

CONCLUSION

The application of microprocessor control on the Dead Reckoning Table was realized at the end of this study.

The errors produced by the operators were prevented and the correct results for the operation problems were obtained with the aid of microprocessor.

In future development, this plotting system can be connected to a central computer on the ship. Therefore, the other weapon control systems can be operated with the data from the DRT.

The most important advantage of this system is the potential for future developments.

BIBLIOGRAPHY

1. LEVENTHAL, L.A., Introduction to Microprocessors, Software, Hardware, Programming, London: Prentice-Hall International Inc., 1980.
2. M 6800 Microprocessor Applications Manuel, Switzerland: Motorola Semiconductor Products Inc., 1975.
3. PEATMAN, J.B., Digital Hardware Design, USA: Mc Graw Hill Publications Co., 1980.
4. M 6800 Microprocessor Programming Manuel, Switzerland: Motorola Semiconductor Products Inc., 1975.
5. HNATEK, E.R., A User's Handbook of D/A and A/D Converters, USA: John Wiley and Sons Inc., 1976.
6. SCOTT, R.N., Electronic Computer Technology, Japan: Mc Graw Hill Book Company, 1970.

7. EKINCI, F., Design of M6800 Based Extendable General Purpose Microcomputer, Master Thesis, Istanbul: Boğaziçi University, 1981.
8. KREYSZIG, E., Advanced Engineering Mathematics, USA: John Wiley and Sons Inc., 1972.
9. OSBORNE, A., An Introduction to Microcomputers, USA: Mc Graw Hill Book Company, 1980.

APPENDIX A PROGRAM LISTING

MAIN PROGRAM

00FD	LDS #0DOO	8E 0D 00	INITIALIZE STACK
0100	LDX #DFOO	CE DF 00	POINTER, INITIALIZE
0103	LDAA #FF	86 FF	PIA'S
0105	LDAB #34	C6 34	
0107	CLR 1,X	6F 01	
0109	STAA 0,X	A7 00	P1ORA OUTPUT PORT
010B	CLR 3,X	6F 03	
010D	STAA 2,X	A7 02	P1ORB OUTPUT PORT
010F	STAB 1,X	E7 01	P1CRA2 CA2 RESET
0111	STAB 3,X	E7 03	P1CRB2 CB2 RESET
0113	CLR 5,X	6F 05	
0115	CLR 4,X	6F 04	P2ORA INPUT PORT
0117	CLR 7,X	6F 07	
0119	CLR 6,X	6F 06	P2ORB INPUT PORT
011B	STAB 5,X	E7 05	P2CRA CA2 RESET
011D	STAB 7,X	E7 07	P2CRB CB2 RESET
011F	CLR 9,X	6F 09	
0121	CLR 8,X	6F 08	P3ORA INPUT PORT
0123	CLR B,X	6F 0B	
0125	CLR A,X	6F 0A	P3ORB INPUT PORT
0127	STAB 9,X	E7 09	P3CRA CA2 RESET
0129	STAB B,X	E7 0B	P3CRB CB2 RESET
012B	CLR D,X	6F 0D	
012D	CLR C,X	6F 0C	P4ORA INPUT PORT
012F	CLR F,X	6F 0F	
0131	CLR E,X	6F 0E	P4ORB INPUT PORT
0133	STAB D,X	E7 0D	P4CRA CA2 RESET
0135	STAB F,X	E7 0F	P4CRB CB2 RESET
0137	LDAA P3ORA	B6 DF 08	READRESET X REF
013A	STAA XTOTH	97 D7	
013C	LDAA P3ORB	B6 DF 0A	READ RESET Y REF
013F	STAA YTOTH	97 DF	
0141	CLR XTOTL	7F 00 D8	
0144	CLR YTOTL	7F 00 E0	
0147	CLR FIRST	7F 00 C0	

014A	BEGIN	LDAA P2ORA	B6 DF 04	READ OWN COURSE
014D		STAA OWNCO	97 B0	
014F		LDAA P2ORB	B6 DF 06	READ OWN SPEED
0152		STAA OWNSP	97 B1	
0154		LDAA P4ORA	B6 DF 0C	READ TARGET BEARING
0157		STAA TARBE	97 B2	
0159		LDAA P4ORB	B6 DF 0E	READ TARGET DISTANCE
015C		STAA TARDI	97 E4	
015E		LDX IN\$CAH	FE 00 C2	READ SCALE
0161		STX SCAHIG	DF C4	
0163		LDAA #9	86 09	MULTIPLY SCALE BY
0165		STAA CARPAN	97 B8	1.125 AND GENERATE
0167		LDX SCAHIG	DE C4	TARGET SCALE
0169		STX CARPLH	DF B9	
016B		LDX #5	CE 00 05	
016E		JSR CARP	BD 02 7E	
0171		LDX CARPMH	DE B6	
0173		STX TARSCH	DF DA	
0175		LDAA OWNCO	96 B0	CONVERT GRAY CODED
0177		JSR GRBIN	BD 02 65	OWN COURSE INTO
017A		LDAA LOCG2	96 B4	BINARY
017C		STAA CARPAN	97 B8	
017E		LDX #B4 00	CE B4 00	
0181		STX CARPLH	DF B9	
0183		LDX #11	CE 00 11	
0186		JSR CARP	BD 02 7E	
0189		LDX CARPMH	DE B6	
018B		STX COURHB	DF C8	
018D		STX OWCBH	DF BB	
018F		LDAA OWNSP	96 B1	
0191		STAA SPD	97 C1	
0193		LDAA TARBE	96 B2	CONVERT GRAY CODED
0195		JSR GRBIN	BD 02 65	TARGET BEARING INTO
0198		LDAA LOCG2	96 B4	BINARY
019A		STAA CARPAN	97 B8	
019C		LDX # B400	CE B4 00	
019F		STX CARPLH	DF B9	
01A1		LDX #11	CE 00 11	
01A4		JSRCARP	BD 02 7E	
01A7		LDX CARPMH	DE B6	
01A9		STX TABEH	DF E5	
01AB	AGAIN	CLR TARGET	7F 00 E2	
01AE		JSR SUB90	BD 02 45	
01B1		LDAB COURHB	D6 C8	BRANCH TO FIRST
01B3		BMI QUAD1	2B 2E	QUADRANT CALCULATION

01B5		JSR SUB90	ED 02 45	IF THE COURSE < 90°
01B8		LDAB COURHB	D6 C8	BRANCH TO FOURTH
01BA		BMI QUAD4	2B 46	QUADRANT CALCULATION
01BC		JSR SUB90	BD 02 45	IF THE COURSE < 180°
01BF		LDAB COURHB	D6 C8	BRANCH TO THIRD
01C1		BMI QUAD3	2B 63	QUADRANT CALCULATION
				IF THE COURSE < 270°
01C3		JSR SINE	BD 02 55	THE COURSE > 270°
01C6		CLR MINUS	7F 00 CE	SECOND QUADRANT
01C9		LDAA #1	86 01	CALCULATION
01CB		STAA VERT	97 CF	
01CD		JSR MULT	BD 02 A3	
01D0		LDAA #5A	86 5A	
01D2		SUBA COURLB	90 C9	
01D4		STAA COURLB	97 C9	
01D6		JSR SINE	BD 02 55	
01D9		LDAA #1	86 01	
01DB		STAA MINUS	97 CE	
01DD		CLR VERT	7F 00 CF	
01E0		JMP MULT	7E 02 A3	
01E3	QUAD1	JSR SINE	BD 02 55	FIND SINE
01E6		CLR MINUS	7F 00 CE	
01E9		LDAA #1	86 01	
01EB		STAA VERT	97 CF	
01ED		JSR MULT	BD 02 A3	
01F0		LDAA #5A	86 5A	
01F2		SUBA COURLB	90 C9	
01F4		STAA COURLB	97 C9	
01F6		JSR SINE	BD 02 55	FIND COSINE
01F9		CLR MINUS	7F 00 CE	
01FC		CLR VERT	7F 00 CF	
01FF		JMP MULT	7E 02 A3	
0202	QUAD4	LDAA #5A	86 5A	
0204		SUBA COURLB	90 C9	
0206		STAA COURLB	97 C9	
0208		JSR SINE	BD 02 55	FIND SINE
020B		LDAA #1	86 01	
020D		STAA MINUS	97 CE	
020F		STAA VERT	97 CF	
0211		JSR MULT	BD 02 A3	
0214		LDAA #5A	86 5A	
0216		SUBA COURLB	90 C9	
0218		STAA COURLB	97 C9	
021A		JSR SINE	BD 02 55	FIND COSINE
021D		CLR MINUS	7F 00 CE	
0220		CLR VERT	7F 00 CF	

0223		JMP MULT	7E 02 A3	
0226	QUAD3	JSR SINE	BD 02 55	FIND SINE
0229		LDAA #1	86 01	
022B		STAA MINUS	97 CE	
022D		STAA VERT	97 CF	
022F		JSR MULT	BD 02 A3	
0232		LDAA #5A	86 5A	
0234		SUBA COURLB	90 C9	
0236		STAA COURLB	97 C9	
0238		JSR SINE	BD 02 55	FIND COSINE
023B		LDAA #1	86 01	
023D		STAA MINUS	97 CE	
023F		CLR VERT	7F 00 CF	
0242		JMP MULT	7E 02 A3	
0245	SUB90	LDAA COURLB	96 C9	SUBTRACT 90° FROM
0247		LDAB COURHB	D6 C8	THE COURSE
0249		SUBA #5A	80 5A	
024B		SBCB #00	C2 00	
024D		BPL POS	2A 01	IF THE COURSE < 90°
024F		NEG A	40	NEGATE IT
0250	POS	STAA COURLB	97 C9	
0252		STAB COURHB	D7 C8	
0254		RTS	39	
0255	SINE	LDAA #08	86 08	USING THE SINE TABLE
0257		STAA SINADH	97 CA	GET THE SINE OF
0259		LDAA COURLB	96 C9	THE ANGLE
0258		ASLA	48	
025C		STAA SINADL	97 CB	
025E		LDX SINADH	DE CA	
0260		LDX 0,X	EE 00	
0262		STX SINHIG	DF CC	
0264		RTS	39	
0265	GRBIN	LDX #7	CE 00 07	CONVERT THE GRAY
0268		ASLA	48	CODED NUMBER INTO
0269		ROL LOCG2	79 00 B4	BINARY. STORE THE
026C	G1	ASLA	48	RESULT IN LOCG2
026D		ROL LOCG1	79 00 B5	
0270		LDAB LOCG1	F6 00 B5	
0273		EORB LOCG2	F8 00 B4	
0276		ASRB	57	
0277		ROL LOCG2	79 00 B4	
027A		DEX	09	
027B		BNE G1	26 EF	
027D		RTS	39	

027E	CARP	CLR CARPMH	7F 00 B6	MULTIPLY THE NUMBER IN CARPM BY CARPLN AND STORE THE RESULT IN CARPM
0281		CLR CARPML	7F 00 B7	
0284		CLC	0C	
0285	CA1	DEX	09	
0286		BEQ CIK	27 1A	
0288		ROR CARPMH	76 00 B6	
028B		ROR CARPML	76 00 B7	
028E		CLC	0C	
028F		ROR CARPAN	76 00 B8	
0292		BCC CA1	24 F1	
0294		LDAA CARPML	96 B7	
0296		LDAB CARPMH	D6 B6	
0298		ADDA CARPLL	9B BA	
029A		ADCB CARPLH	D9 B9	
029C		STAA CARPML	97 B7	
029E		STAB CARPMH	D7 B6	
02A0		BRA CA1	20 E3	
02A2	CIK	RTS	39	
02A3	MULT	CLRA	4F	CLEAR THE REGISTERS FOR MULTIPLYING SINE BY SCALE. SET THE SHIFT COUNT TO \$ 12 DECREMENT SHIFT COUNT IF THE SHIFT COUNT IS EQUAL TO ZERO GOTO THEN SHIFT RIGHT RESULT SHIFT RIGHT SINE IF CARRY CLEAR GOTO NEW ADD SCALE TO RESULT
02A4		STAA RESH	97 D0	
02A6		STAA RESL	97 D1	
02A8		LDX #0012	CE 00 12	
02AB		CLC	0C	
02AC	NEW	DEX	09	
02AD		BEQ THEN	27 1D	
02AF		ROR RESH	76 00 D0	
02B2		ROR RESL	76 00 D1	
02B5		CLC	0C	
02B6		ROR SINHIGH	76 00 CC	
02B9		ROR SINLOW	76 00 CD	
02BC		BCC NEW	24 EE	
02BE		LDAA RESL	96 D1	
02C0		LDAB RESH	D6 D0	
02C2		ADDA SCALOW	9B C5	
02C4		ADCB SCAHIG	D9 C4	
02C6		STAA RESL	97 D1	
02C8		STAB RESH	D7 D0	
02CA		BRA NEW	20 E0	
02CC	THEN	LDAA SPD	96 C1	MULTIPLY SPEED BY RESULT WHICH IS SCALED SINE
02CE		STAA CARPAN	97 B8	
02D0		LDX RESH	DE D0	
02D2		STX CARPLH	DF B9	
02D4		LDX #0A	CE 00 0A	
02D7		JSR CARP	BD 02 7E	
02DA		LDX CARPMH	DE B6	

02DC		STX NETH	DF D2	STORE THE RESULT IN NET
02DE		LDAA MINUS	96 CE	GET SIGN AND RESULT
02EO		LDX NETH	DE D2	
02E2		LDAB VERT	D6 CF	
02E4		BEQ BC	27 OF	
02E6		LDAB TARGET	D6 E2	
02E8		BNE BA	26 06	
02EA		STX YPOSH	DF DC	STORE THEM FOR OWN SHIP Y AXIS
02EC		STAA SGYPOS	97 DE	
02EE		BRA BB	20 04	
02FO	BA	STX YTARH	DF A8	STORE THEM FOR TARGET Y AXIS
02F2		STAA SGYTAR	97 EE	RETURN FOR CALCULATION ON X AXIS
02F4	BB	RTS	39	
02F5	BC	LDAB TARGET	D6 E2	
02F7		BNE CA	26 06	
02F9		STX XPOSH	DF D4	STORE RESULTS FOR OWN SHIP X AXIS
02FB		STAA SGXPOS	97 D6	
02FD		BRA CC	20 06	
02FF	CA	STX XTARH	DF A6	STORE RESULTS FOR TARGET X AXIS
0301		STAA SGXTAR	97 E9	
0303		BRA CD	20 13	
0305	CC	LDAA TARDI	96 E4	GET THE TARGET INFORMATION
0307		STAA SPD	97 C1	
0309		LDX TARSCH	DE DA	
030B		STX SCAHIG	DF C4	
030D		LDX TABEH	DE E5	
030F		STX COURHB	DF C8	
0311		LDAA *FF	86 FF	
0313		STAA TARGET	97 E2	SET TARGET FLAG
0315		JMP AGAIN	7E 01 AE	JUMP TO AGAIN FOR TARGET CALCULATION
0318	CD	LDAA #3	86 03	SHIFT LEFT TARGET COMPONENTS THREE TIMES
031A	LOOP	ASL XTARL	78 00 A7	
031D		ROL XTARH	79 00 A6	
0320		ASL YTARL	78 00 A9	
0323		ROL YTARH -	79 00 A8	
0326		DEC A	4A	
0327		BNE LOOP	26 F1	
0329		LDX XTARH	DE A6	
032B		STX XTARHC	DF E7	
032D		LDX YTARH	DE A8	
032F		STX YTARHC	DF EC	
0331		CLI	0E	CLEAR INT MASK
0332		JMP BEGIN	7E 01 4A	JUMP TO THE BEGINNING

INTERRUPT SERVICE ROUTINE

0400	LDX # 08B6	CE 08 B6	
0403	JSR NAME	BD 06 75	PRINT "OWN"
0406	LDX # 08C7	CE 08 C7	
0409	JSR NAME	BD 06 75	PRINT "COURSE"
040C	LDX OWCBH	DE BB	
040E	STX HB	DF BD	
0410	JSR WRI	BD 06 80	PRINT OWN COURSE
0413	LDX # 08B6	CE 08 B6	
0416	JSR NAME	BD 06 75	PRINT "OWN"
0419	LDX # 08CF	CE 08 CF	
041C	JSR NAME	BD 06 75	PRINT "SPEED"
041F	LDAA OWNSP	96 B1	
0421	STAA LB	97 BE	
0423	CLR HB	7F 00 BD	
0426	JSR WRI	BD 06 80	PRINT OWN SPEED
0429	LDAA # 34	86 34	RESET THE FLAG
042B	STAA CONT	97 BF	
042D	LDX # 00D4	CE 00 D4	
0430	JSR FPOS	BD 04 7E	FIND OWN SHIP X
0433	LDX # 00DC	CE 00 DC	DEPARTURE
0436	JSR FPOS	BD 04 7E	FIND OWN SHIP Y
0439	LDAA CONT	96 BF	DEPARTURE
043B	STAA P1CRB	B7 DF 03	
043E	BITA # 08	85 08	IF THE POSITION
0440	BNE CHANGE	26 3B	OUTSIDE GOTO CHANGE
0442	JSR PLOTME	BD 04 A7	PLOT OWN SHIP
0445	LDX XTOTH	DE D7	GET THE OWN SHIP
0447	STX YTH	DF EA	POSITION AS REFERENCE
0449	LDX YTOTH	DE DF	FOR TARGET
044B	STX YTH	DF EF	
044D	LDX # 00E7	CE 00 E7	
0450	JSR FPOS	BD 04 7E	FIND THE TARGET
0453	LDX # 00EC	CE 00 EC	DISTANCE ON X AXIS
0456	JSR FPOS	BD 04 7E	FIND THE TARGET
0459	TST TARDI	7D 00 E4	DISTANCE ON Y AXIS
045C	BEQ CHANGE	27 1F	IF THE TARGET DISTANCE
045E	LDAA CONT	96 BF	IS ZERO GOTO CHANGE
0460	STAA P2CRA	B7 DF 05	
0463	BITA # 08	85 08	IF THE TARGET POSITION
0465	BNE CHANGE	26 16	IS NOT PLOTTABLE GOTO
			CHANGE
0467	LDAA FIRST	96 C0	IF THIS IS NOT THE
0469	BEQ 1 1	27 03	FIRST PASS DO NOT
046B	JSR CAL	BD 05 28	CALCULATE TARGET

046E	I1	JSR PLOTAR	BD 04	F7	PLOT THE TARGET
0471		LDX YTH	DE EA		
0473		STX XTTH1	DF 94		STORE THE TARGET
0475		LDX YTH	DE EF		POSITION FOR SECOND
0477		STX YTH1	DF 96		PASS CALCULATIONS
0479		LDAA #FF	86 FF		SET FIRST FLAG
047B		STAA FIRST	97 CO		
047D	CHANGE	RTI	3B		SERVICE ROUTINE IS FINISHED
047E	FPOS	LDAA 4,X	A6 04		FIND THE LAST POSITION
0480		LDAB 3,X	E6 03		
0482		TST 2,X	6D 02		
0484		BNE F1	26 08		IF THE MOTION IS
0486		ADDA 1,X	AB 01		POSITIVE ADD THE
0488		ADCB 0,X	E9 00		DEPARTURE TO THE
048A		BCS F3	25 15		LAST POSITION
048C		BRA F2	20 06		
048E	F1	SUBA 1,X	A0 01		IF THE MOTION IS
0490		SBCB 0,X	E2 00		NEGATIVE SUBTRACT
0492		BCS F3	25 0D		THE DEPARTURE FROM
0494	F2	STAA 4,X	A7 04		LAST LOCATION
0496		STAB 3,X	E7 03		
0498		ADDB #05	CB 05		TEST IF THE POSITION
049A		BCS F3	25 05		IS PLOTTABLE
049C		SUBB #0A	C0 0A		
049E		BCS F3	25 01		
04A0		RTS	39		
04A1	F3	LDAA 3C	86 3C		IF NOT, SET THE FLAG
04A3		STAA CONT	B7 00	BF	
04A6		RTS	39		
04A7	PLOTME	LDAA XTOTH	96 D7		
04A9		LDAB YTOTH	D6 DF		
04AB		JSR STORE	BD 04	DO	BRING THE PEN TO THE
04AE		JSR DELAY	BD 04	DA	OWN SHIP POSITION
04B1		LDAA #3C	86 3C		
04B3		STAA PICRA	B7 DF	01	DROP THE PEN
04B6		JSR DELAY	BD 04	DA	
04B9		LDAA XTOTH	96 D7		
04BB		ADDA #5	8B 05		DRAW A SQUARE
04BD		JSR STORE	BD 04	DO	ONTO THE POSITION
04C0		ADDB #5	CB 05		
04C2		JSR STORE	BD 04	DO	
04C5		LDAA XTOTH	96 D7		
04C7		JSR STORE	BD 04	DO	
04CA		LDAB YTOTH	D6 DF		
04CC		JSR CLEAR	BD 04	E3	BRING THE PEN TO THE
04CF		RTS	39		STARTING POINT

04D0	STORE	STAA PLORA	B7 DF 00	OUTPUT THE CONTENTS
04D3		STAB PLORB	F7 DF 02	OF ACCUMULATORS
04D6		JSR DELAY	BD 04 DA	WAIT FOR THE RESPONSE
04D9		RTS	39	OF THE PEN
04DA	DELAY	LDX # 8000	CE 80 00	DELAY SUBROUTINE
04DD	L1	PSHA	36	
04DE		PULA	32	
04DF		DEX	09	
04E0		BNE L1	26 FB	
04E2		RTS	39	
04E3	CLEAR	JSR STORE	BD 04 DO	CLEAR THE OUTPUT
04E6		LDAA # 34	86 34	REGISTERS
04E8		STAA PICRA	B7 DF 01	
04EB		JSR DELAY	BD 04 DA	
04EE		CLRA	4F	
04EF		CLRB	5F	
04F0		JSR STORE	BD 04 DO	
04F3		JSR DELAY	BD 04 DA	
04F6		RTS	39	
04F7	PLOTAR	LDAA XTTH	96 EA	
04F9		LDAB YTTH	D6 EF	
04FB		JSR STORE	BD 04 DO	BRING THE PEN TO
04FE		JSR DELAY	BD 04 DA	THE TARGET POSITION
0501		LDAA # 3C	86 3C	
0503		STAA PICRA	B7 DF 01	DROP THE PEN
0506		JSR DELAY	BD 04 DA	
0509		LDAA XTTH	96 EA	
050B		ADDA # 3	8B 03	DRAW A DIAMOND SHAPE
050D		ADDB # 3	CB 03	ONTO THE TARGET
050F		JSR STORE	BD 04 DO	POSITION
0512		LDAA XTTH	96 EA	
0514		ADDB # 3	CB 03	
0516		JSR STORE	BD 04 DO	
0519		SUBA # 3	80 03	
051B		SUBB # 3	C0 03	
051D		JSR STORE	BD 04 DO	
0520		LDAA XTTH	96 EA	
0522		LDAB YTTH	D6 EF	
0524		JSR CLEAR	BD 04 E3	BRING THE PEN
0527		RTS	39	TO THE STARTING POINT
0528	CAL	LDX XTTH	DE EA	FIND THE X DEPARTURE
052A		STX LOCD3	DF 9D	OF THE TARGET
052C		LDX XTTH1	DE 94	
052E		STX LOCD1	DF 9B	

0530		JSR DIFF	BD 06 E9	
0533		LDAA LOCNEG	96 9F	
0535		STAA XSIGN	97 9A	
0537		LDAA IOCDE	96 9B	
0539		STAA XTADEP	97 A3	
053B		JSR SQUARE	BD 07 5C	SQUARE X DEP
053E		LDX CARPMH	DE B6	
0540		STX SQUH	DF 98	
0542		LDX YPTH	DE EF	
0544		STX IOCD3	DF 9D	
0546		LDX YPTH1	DE 96	
0548		STX IOCD1	DF 9B	
054A		JSR DIFF	BD 06 E9	FIND Y DEPARTURE
054D		LDAA LOCNEG	96 9F	
054F		STAA YSIGN	97 A5	
0551		LDAA IOCD1	96 9B	
0553		STAA YTADEP	97 A4	
0555		JSR SQUARE	BD 07 5C	SQUARE IT
0558		LDAA CARPML	96 B7	
055A		LDAB CARPMH	D6 B6	
055C		ADDA SQUH	9B 99	ADD X ² TO Y ²
055E		ADCB SQUH	D9 98	
0560		STAA SQUH	97 99	
0562		STAB SQUH	D7 98	
0564		LDAA # 1	86 01	FIND THE SQUARE ROOT
0566		CMPB # 64	C1 40	OF (X ² +Y ²) BY
0568		BCC KAR1	24 0A	NEWTON-RAPHSON METHOD
056A		CMPB # 16	C1 10	IN ORDER TO FIND
056C		BCC KAR2	24 08	THE DEPARTURE
056E		CMPB # 1	C1 01	
0570		BCC KAR3	24 06	
0572		BRA KAR4	20 06	
0574	KAR1	ADDA # 40	8B 40	
0576	KAR2	ADDA # 30	8B 30	
0578	KAR3	ADDA # OF	8B OF	
057A	KAR4	LOAB # 0A	C6 0A	
057C	KAR5	STAA TEMPL	97 C7	
057E		STAA DVSR	97 AA	
0580		LDX SQUH	DE 98	
0582		BEQ KAR6	27 16	
0584		STX DVDNH	DF AB	
0586		PSHB	37	
0587		JSR DIV	BD 07 09	
058A		LDAB QUOTH	D6 AD	
058C		LDAA QUOTL	96 AE	
058E		ADDA TEMPL	9B C7	
0590		ADCB # 00	C9 00	

0592		LSR B	54		
0593		ROR A	46		
0594		PUL B	33		
0595		DEC B	5A		
0596		BNE KAR5	26 F4		
0598		BRA KAR7	20 01		
059A	KAR6	CLRA	4F		
059B	KAR7	STAA DEP	97 B3		
059D		STAA DVDNH	97 AB		
059F		CLR DVDNL	7F 00 AC		
05A2		LDAA INSCAH	96 C2		
05A4		STAA DVSR	97 AA		
05A6		JSR DIV	BD 07 09		DIVIDE THE DEPARTURE
05A9		LDAA QUOTL	96 AE		TO THE SCALE TO
05AB		STAA TARSPD	97 A2		FIND THE SPEED OF
05AD		LDAA DEF	96 B3		THE TARGET
05AF		BEQ NOCO	27 43		IF THE DEPARTURE IS
05B1		STAA DVSR	97 AA		ZERO THE TARGET HAS
05B3		LDAA YTADEP	96 A4		NO SPEED AND COURSE
05B5		STAA DVDNH	97 AB		
05B7		CLR DVDNL	7F 00 AC		
05BA		JSR DIV	BD 07 09		IF NOT DIVIDE Y
05BD		LDAA QUOTH	96 AD		DEPARTURE TO THE
05BF		BEQ RO1	27 04		DEPARTURE TO FIND
05C1		LDAA # 5A	86 5A		THE SINE OF THE
05C3		BRA NIN	20 10		ANGLE (COURSE)
05C5	RO1	LDAA QUOTL	96 AE		
05C7		LDX #07FE	CE 07 FE		
05CA	RO2	INX	08		
05CB		INX	08		
05CC		CMPA 0,X	A1 00		COMPARE THE SINE
05CE		BHI RO2	22 FA		WITH THE TERMS ON THE
05D0		STX TEMPH	DF C6		LOOK UP TABLE
05D2		LDAA TEMPL	96 C7		
05D4		LSRA	44		
05D5	NIN	STAA TEMPE-	97 C7		
05D7		LDAB # 5A	C6 5A		
05D9		LDAA XSIGN	96 9A		CALCULATE THE TARGET
05DB		EORA YSIGN	98 A5		COURSE
05DD		BEQ RO3	27 04		
05DF		ADDB TEMPL	DB C7		
05E1		BRA RO4	20 02		
05E3	RO3	SUBB TEMPL	DO C7		
05E5	RO4	LDAA XSIGN	96 9A		
05E7		BEQ RO5	27 05		
05E9		CLRA	4F		
05EA		ADDB #B4	CB B4		
05EC		ADCA #00	89 00		

05EE	RO5	STAA TARCOH	97	AO	
05FO		STAB TARCOL	D7	A1	
05F2		BRA YAZ	20	05	
05F4	NOCO	LDX # FFFF	CE	FF	FF
05F7		STX TARCOH	DF	AO	
					TAKE \$ FFFF AS TARGET COURSE FOR PRINTING "???"
05F9	YAZ	LDX #08BD	CE	08	BD
05FC		JSR NAME	BD	06	75
05FF		LDX #08D6	CE	08	D6
0602		JSR NAME	BD	06	75
0605		LDX TABEH	DE	E5	
0607		STX HB	DF	BD	
0609		JSR WRI	BD	06	80
060C		LDX #08BD	CE	08	BD
060F		JSR NAME	BD	06	75
0612		LDX #08DF	CE	08	DF
0615		JSR NAME	BD	06	75
0618		LDAA TARDI	96	E4	
061A		STAA LB	97	BE	
061C		CLR HB	7F	00	BD
061F		JSR WRI	BD	06	80
0622		LDAA #30	86	30	
0624		JSR OUT	BD	06	6A
0627		JSR OUT	BD	06	6A
					PRINT "TARGET"
					PRINT "BEARING"
					PRINT "TARGET BEARING"
					PRINT "TARGET"
					PRINT "DISTANCE"
					PRINT "TARGET DIS- TANCE"
062A		LDX #08BD	CE	08	BD
062D		JSR NAME	BD	06	75
0630		LDX #08C7	CE	08	C7
0633		JSR NAME	BD	06	75
0636		LDX TARCOH	DE	AO	
0638		STX HB	DF	BD	
063A		BPL YA1	2A	07	
063C		LDAA HB	96	BD	
063E		JSR THEX	BD	06	BC
0641		BRA YA2	20	03	
0643	YA1	JSR WRI	BD	06	80
0646	YA2	LDX #08BD	CE	08	BD
0649		JSR NAME	BD	06	75
064C		LDX #08CF	CE	08	CF
064F		JSR NAME	BD	06	75
0652		LDAA TARSPD	96	A2	
0654		STAA LB	97	BE	
0656		CLR HB	7F	00	BD
0659		JSR WRI	BD	06	80
065C		LDAA #0D	86	0D	
065E		JSR OUT	BD	06	6A
0661		LDAA #0A	86	0A	
0663		JSR OUT	BD	06	6A
					PRINT TARGET COURSE
					PRINT "TARGET"
					PRINT "SPEED"
					PRINT TARGET SPEED TRANSMIT OR AND TWO LF

0666		JSR OUT	BD 06 6A	
0669		RTS	39	
066A	OUT	LDAB ACIACR	F6 DF 10	
066D		ASRB	57	
066E		ASRB	57	
066F		BCC OUT	24 F9	IF TX DATA REG IS
0671		STAA ACIADR	B7 DF 11	EMPTY, OUTPUT THE
0674		RTS	39	CHARACTER
0675	NAME	LDAA O,X	A6 00	GET THE CHARACTER
0677		BEQ NA1	27 06	FROM THE STRING
0679		JSR OUT	BD 06 6A	IF IT IS NOT ZERO
067C		INX	08	TRANSMIT IT
067D		BRA NAME	20 F6	GET NEW CHARACTER
067F	NA1	RTS	39	IF IT IS ZERO RETURN
				FROM SUBROUTINE
0680	WRI	LDAA LB	96 BE	CONVERT THE BINARY
0682		LDAB HB	D6 BD	NUMBER BETWEEN 0-359
0684		SUBA # 64	80 64	TO BCD
0686		SBCB # 00	C2 00	
0688		BPL WR1	2A 07	
068A		ADDA # 64	8B 64	
068C		JSR BCD	BD 06 DA	
068F		BRA WR4	20 29	
0691	WR1	SUBA # 64	80 64	
0693		SBCB # 00	C2 00	
0695		BPL WR2	2A 0B	
0697		ADDA # 64	8B 64	
0699		JSR BCD	BD 06 DA	
069C		ADDB # 9C	CB 9C	
069E		ADCA # 00	89 00	
06A0		BRA WR4	20 18	
06A2	WR2	SUBA # 64	80 64	
06A4		SBCB # 00	C2 00	
06A6		BPL WR3	2A 0B	
06A8		ADDA # 64	8B 64	
06AA		JSR BCD	BD 06 DA	
06AD		ADDB # 38	CB 38	
06AF		ADCA # 01	89 01	
06B1		BRA WR4	20 07	
06B3	WR3	JSR BCD	BD 06 DA	
06B6		ADDB # D4	CB D4	
06B8		ADCA # 01	89 01	
06BA	WR4	STAB LB	D7 BE	
06BC	THEX	ANDA # 0F	84 0F	TRANSMIT THREE DIGITS
06BE		ADDA # 30	8B 30	
06C0		JSR OUT	BD 06 DA	
06C3		LDAA LB	96 BE	
06C5		ANDA # 0F	84 0F	

06C7		LSRA	44	
06C8		LSRA	44	
06C9		LSRA	44	
06CA		LSRA	44	
06CB		ADDA # 30	8B 30	
06CD		JSR OUT	BD 06 6A	
06D0		LDAA LB	96 BE	
06D2		ANDA # 0F	84 0F	
06D4		ADDA # 30	8B 30	
06D6		JSR OUT	BD 06 6A	
06D9		RTS	39	
06DA	BCD	CLR B	5F	CONVERT THE BINARY NUMBER BETWEEN 0-100 INTO BCD
06DB	DEV	SUBA # 0A	80 0A	
06DD		BMI SON	2B 04	
06DF		ADDB # 06	CB 06	
06E1		BRA DEV	20 F8	
06E3	SON	CLR A	4F	
06E4		ADDB LB	DB BE	
06E6		ADCA HB	99 BD	
06E8		RTS	39	
06E9	DIFF	CLR IOCNEG	7F 00 9F	DOUBLE PRECISION SUBTRACTION
06EC		LDAA LOCD4	96 9E	
06EE		LDAB LOCD3	D6 9D	
06F0		SUBA LOCD2	90 9C	
06F2		SBCB LOCD1	D2 9B	
06F4		BCC DA1	24 09	
06F6		COM A	43	
06F7		COM B	53	
06F8		ADDA # 01	8B 01	
06FA		ADCB # 00	C9 00	
06FC		COM LOCNEG	73 00 9F	
06FF	DA1	STAB LOCD1	D7 9B	
0701		STAA E OCD2	97 9C	
0703		EPL DA2	2A 03	
0705		INC LOCD1	7C 00 9B	
0708	DA2	RTS	39	
0709	DIV	LDAB # 08	06 08	DIVIDE THE NUMBER IN DVDNH, DVDNL BY DVSR AND STORE THE RESULT IN QUOTH, QUOTL
070B		CLR QUOTH	7F 00 AD	
070E		CLR QUOTL	7F 00 AE	
0711	DIV2	INCB	5C	
0712		CMPB # 16	C1 10	
0714		BGT DIV1	2E 34	
0716		ASL DVSR	78 00 AA	
0719		BCC DIV2	24 F6	
071B		STAB SHCO	D7 AF	

071D		ROR DVSR	76	00	AA	
0720		LDAA DVDNH	96	AB		
0722	DIV7	CMPA DVSR	91	AA		
0724		BCS DIV3	25	0D		
0726	DIV6	SEC	0D			
0727		ROL QUOTL	79	00	AE	
072A		ROL QUOTH	79	00	AD	
072D		SUBA DVSR	90	AA		
072F		STAA DVDNH	97	AB		
0731		BRA DIV4	20	07		
0733	DIV3	CLC	0C			
0734		ROL QUOTL	79	00	AE	
3737		ROL QUOTH	79	00	AD	
073A	DIV4	DECB	5A			
073B		BEQ DIV5	27	12		
073D		CLC	0C			
073E		ROL DVDNL	79	00	AC	
0741		ROL DVDNH	79	00	AB	
0744		LDAA DVDNH	96	AB		
0746		BCS DIV6	25	DE		
0748		BRA DIV7	20	D8		
074A	DIV1	LDX #FFFF	CE	FF	FF	
074D		STX QUOTH	DF	AD		
074F	DIV5	LDAB SHCO	D6	AF		
0751		SUBB #9	C0	09		
0753		CMPB #4	C1	04		
0755		BCS DIV8	25	02		
0757		SUBB #4	C0	04		
0759	DIV8	STAB SHCO	D7	AF		
075B		RTS	39			
075C	SQUARE	STAA CARPLH	97	B9		
075E		STAA CARPAN	97	B8		
0760		CLR CARPLL	7F	00	BA	
0763		LDX #0A	CE	00	0A	
0766		JSR CARP	FD	02	7E	
0769		RTS	39			

CALCULATE THE
SQUARE OF THE
NUMBER IN ACC A

SINE TABLE

0800	00	00	04	78	08	EF	0D	66	11	DB	16	50	1A	C2	1F	33	0°- 7°
0810	23	A1	28	0C	2C	74	30	D9	35	3A	39	96	3D	EE	42	42	8°-15°
0820	46	90	4A	D9	4F	1B	53	58	57	8E	5B	BE	5F	E6	64	07	16°-23°
0830	68	20	6C	30	70	39	74	38	78	2F	7C	1C	80	01	83	D9	24°-31°
0840	87	A8	8B	6D	8F	27	92	D5	96	78	9A	10	9D	9B	A1	1A	32°-39°
0850	A4	8D	A7	F3	AB	4C	AE	97	B1	D4	B5	04	B8	26	BB	39	40°-47°
0860	BE	3E	C1	34	C4	1A	C6	F2	C9	BA	CC	73	CF	1B	D1	B3	48°-55°
0870	D4	3B	D6	B2	D9	18	DB	6E	DD	B3	DF	E6	E2	08	E4	18	56°-63°
0880	E6	16	E8	03	E9	DD	EB	A5	ED	5B	EE	FE	FO	8F	F2	0D	64°-71°
0890	F3	78	F4	CF	F6	14	F7	46	F8	64	F9	6F	FA	67	FB	4B	72°-79°
08A0	FC	1B	FC	D8	FD	81	FE	16	FE	98	FF	06	FF	5F	FF	A5	80°-87°
08B0	FF	D7	FF	F5	FF	FF											88°-90°

CHARACTER TABLE

08B6	0D	0A	4F	57	4E	20	00										CR LF OWN SP NUL
08BD	0D	0A	54	41	52	47	45	54	20	00							CR LF TARGET SP NUL
08C7	43	4F	55	52	53	45	3A	00									COURSE : NUL
08CF	53	50	45	45	44	3A	00										SPEED : NUL
08D6	42	45	41	52	49	4E	47	3A	00								BEARING : NUL
08DF	44	49	53	54	41	4E	43	45	3A	00							DISTANCE : NUL

APPENDIX B MPU INSTRUCTIONS

Accumulator and Memory Instructions

ADDRESSING MODES

BOOLEAN/ARITHMETIC OPERATION

COND. CODE REG.

OPERATIONS	MNEMONIC	ADDRESSING MODES					(All register labels refer to contents)	COND. CODE REG.							
		IMMED	DIRECT	INDEX	EXTND	IMPLIED		H	I	N	Z	V	C		
		OP ~ #	OP ~ #	OP ~ #	OP ~ #	OP ~ #									
Add	ADDA	8B 2 2	9B 3 2	AB 5 2	BB 4 3		A ← M → A	↑	↑	↑	↑	↑	↑	↑	↑
	ADDB	CB 2 2	DB 3 2	EB 5 2	FB 4 3		B ← M → B	↑	↑	↑	↑	↑	↑	↑	↑
Add Acmltrs	ABA					1B 2 1	A ← B → A	↑	↑	↑	↑	↑	↑	↑	↑
Add with Carry	ADCA	89 2 2	99 3 2	A9 5 2	89 4 3		A ← M ← C → A	↑	↑	↑	↑	↑	↑	↑	↑
	ADCB	C9 2 2	D9 3 2	E9 5 2	F9 4 3		B ← M ← C → B	↑	↑	↑	↑	↑	↑	↑	↑
And	ANDA	84 2 2	94 3 2	A4 5 2	B4 4 3		A ← M → A	↑	↑	↑	↑	↑	↑	↑	↑
	ANDB	C4 2 2	D4 3 2	E4 5 2	F4 4 3		B ← M → B	↑	↑	↑	↑	↑	↑	↑	↑
Bit Test	BITA	85 2 2	95 3 2	A5 5 2	B5 4 3		A ← M	↑	↑	↑	↑	↑	↑	↑	↑
	BITB	C5 2 2	D5 3 2	E5 5 2	F5 4 3		B ← M	↑	↑	↑	↑	↑	↑	↑	↑
Clear	CLR			6F 7 2	7F 6 3		00 → M	↑	↑	↑	↑	↑	↑	↑	↑
	CLRA					4F 2 1	00 → A	↑	↑	↑	↑	↑	↑	↑	↑
	CLRB					5F 2 1	00 → B	↑	↑	↑	↑	↑	↑	↑	↑
Compare	CMPA	81 2 2	91 3 2	A1 5 2	81 4 3		A ← M	↑	↑	↑	↑	↑	↑	↑	↑
	CMPB	C1 2 2	D1 3 2	E1 5 2	F1 4 3		B ← M	↑	↑	↑	↑	↑	↑	↑	↑
Compare Acmltrs	CBA					11 2 1	A ← B	↑	↑	↑	↑	↑	↑	↑	↑
	COM			83 7 2	73 6 3		M ← M	↑	↑	↑	↑	↑	↑	↑	↑
Complement, 1's	COMA					43 2 1	X ← A	↑	↑	↑	↑	↑	↑	↑	↑
	COMB					53 2 1	B ← B	↑	↑	↑	↑	↑	↑	↑	↑
Complement, 2's (Negate)	NEG			60 7 2	70 6 3		00 ← M → M	↑	↑	↑	↑	↑	↑	↑	↑
	NEGA					40 2 1	00 ← A → A	↑	↑	↑	↑	↑	↑	↑	↑
Decimal Adjust, A	NEGB					50 2 1	00 ← B → B	↑	↑	↑	↑	↑	↑	↑	↑
	DAA					19 2 1	Converts Binary Add. of BCD Characters into BCD Format	↑	↑	↑	↑	↑	↑	↑	↑
Decrement	DEC			6A 7 2	7A 6 3		M ← 1 → M	↑	↑	↑	↑	↑	↑	↑	↑
	DECA					4A 2 1	A ← 1 → A	↑	↑	↑	↑	↑	↑	↑	↑
Exclusive OR	DECB					5A 2 1	B ← 1 → B	↑	↑	↑	↑	↑	↑	↑	↑
	EORA	88 2 2	98 3 2	A8 5 2	B8 4 3		A ⊕ M → A	↑	↑	↑	↑	↑	↑	↑	↑
Increment	EORB	C8 2 2	D8 3 2	E8 5 2	F8 4 3		B ⊕ M → B	↑	↑	↑	↑	↑	↑	↑	↑
	INC			6C 7 2	7C 6 3		M ← 1 → M	↑	↑	↑	↑	↑	↑	↑	↑
Load Acmltr	INCA					4C 2 1	A ← 1 → A	↑	↑	↑	↑	↑	↑	↑	↑
	INCB					5C 2 1	B ← 1 → B	↑	↑	↑	↑	↑	↑	↑	↑
Or, Inclusive	LDAA	86 2 2	96 3 2	A6 5 2	86 4 3		M ← A	↑	↑	↑	↑	↑	↑	↑	↑
	LDAB	C6 2 2	D6 3 2	E6 5 2	F6 4 3		M ← B	↑	↑	↑	↑	↑	↑	↑	↑
Push Data	ORAA	8A 2 2	9A 3 2	AA 5 2	BA 4 3		A ← M → A	↑	↑	↑	↑	↑	↑	↑	↑
	ORAB	CA 2 2	DA 3 2	EA 5 2	FA 4 3		B ← M → B	↑	↑	↑	↑	↑	↑	↑	↑
Pull Data	PSHA					36 4 1	A ← M _{SP} , SP ← 1 → SP	↑	↑	↑	↑	↑	↑	↑	↑
	PSHB					37 4 1	B ← M _{SP} , SP ← 1 → SP	↑	↑	↑	↑	↑	↑	↑	↑
Rotate Left	PULA					32 4 1	SP ← 1 → SP, M _{SP} → A	↑	↑	↑	↑	↑	↑	↑	↑
	PULB					33 4 1	SP ← 1 → SP, M _{SP} → B	↑	↑	↑	↑	↑	↑	↑	↑
Rotate Right	ROL			69 7 2	79 6 3		M	↑	↑	↑	↑	↑	↑	↑	↑
	ROLA					49 2 1	A	↑	↑	↑	↑	↑	↑	↑	↑
Shift Left, Arithmetic	ROLB					59 2 1	B	↑	↑	↑	↑	↑	↑	↑	↑
	ROR			66 7 2	76 6 3		M	↑	↑	↑	↑	↑	↑	↑	↑
Shift Right, Arithmetic	RORA					46 2 1	A	↑	↑	↑	↑	↑	↑	↑	↑
	RORB					56 2 1	B	↑	↑	↑	↑	↑	↑	↑	↑
Shift Right, Logic	ASL			68 7 2	78 6 3		M	↑	↑	↑	↑	↑	↑	↑	↑
	ASLA					48 2 1	A	↑	↑	↑	↑	↑	↑	↑	↑
Store Acmltr.	ASLB					68 2 1	B	↑	↑	↑	↑	↑	↑	↑	↑
	ASR			67 7 2	77 6 3		M	↑	↑	↑	↑	↑	↑	↑	↑
Shift Right, Arithmetic	ASRA					47 2 1	A	↑	↑	↑	↑	↑	↑	↑	↑
	ASRB					57 2 1	B	↑	↑	↑	↑	↑	↑	↑	↑
Store Acmltr.	LSR			64 7 2	74 6 3		M	↑	↑	↑	↑	↑	↑	↑	↑
	LSRA					44 2 1	A	↑	↑	↑	↑	↑	↑	↑	↑
Subtract	LSRB					54 2 1	B	↑	↑	↑	↑	↑	↑	↑	↑
	STAA		97 4 2	A7 6 2	87 5 3		A ← M	↑	↑	↑	↑	↑	↑	↑	↑
Subtract Acmltrs.	STAB		D7 4 2	E7 6 2	F7 5 3		B ← M	↑	↑	↑	↑	↑	↑	↑	↑
	SUBA	80 2 2	90 3 2	A0 5 2	80 4 3		A ← M → A	↑	↑	↑	↑	↑	↑	↑	↑
Subtr. with Carry	SUBB	C0 2 2	D0 3 2	E0 5 2	F0 4 3		B ← M → B	↑	↑	↑	↑	↑	↑	↑	↑
	SBA					10 2 1	A ← B → A	↑	↑	↑	↑	↑	↑	↑	↑
Transfer Acmltrs	SBCA	82 2 2	92 3 2	A2 5 2	82 4 3		A ← M ← C → A	↑	↑	↑	↑	↑	↑	↑	↑
	SBCB	C2 2 2	D2 3 2	E2 5 2	F2 4 3		B ← M ← C → B	↑	↑	↑	↑	↑	↑	↑	↑
Test, Zero or Minus	TAB					16 2 1	A ← B	↑	↑	↑	↑	↑	↑	↑	↑
	TBA					17 2 1	B ← A	↑	↑	↑	↑	↑	↑	↑	↑
TST	TST			6D 7 2	7D 6 3		M ← 00	↑	↑	↑	↑	↑	↑	↑	↑
	TSTA					4D 2 1	A ← 00	↑	↑	↑	↑	↑	↑	↑	↑
	TSTB					5D 2 1	B ← 00	↑	↑	↑	↑	↑	↑	↑	↑

LEGEND:
 OP Operation Code (Hexadecimal);
 ~ Number of MPU Cycles;
 # Number of Program Bytes;
 + Arithmetic Plus;
 - Arithmetic Minus;
 & Boolean AND;
 M_{SP} Contents of memory location pointed to by Stack Pointer;

+ Boolean Inclusive OR;
 ⊕ Boolean Exclusive OR;
 M Complement of M;
 ← Transfer Into;
 0 Bit = Zero;
 00 Byte = Zero;

CONDITION CODE SYMBOLS:
 H Half-carry from bit 3;
 I Interrupt mask;
 N Negative (sign bit);
 Z Zero (byte);
 V Overflow, 2's complement;
 C Carry from bit 7;
 R Reset Always;
 S Set Always;
 † Test and set if true, cleared otherwise;
 • Not Affected

Note - Accumulator addressing mode instructions are included in the column for IMPLIED addressing

Index Register and Stack Manipulation Instructions

		BOOLEAN/ARITHMETIC OPERATION												COND. CODE REG.									
		IMMED			DIRECT			INDEX			EXTND			IMPLIED									
POINTER OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	5	4	3	2	1	0	
Compare Index Reg	CPX	BC	3	3	9C	4	2	AC	6	2	BC	5	3										
Decrement Index Reg	DEX													09	4	1							
Decrement Stack Ptr	DES													34	4	1							
Increment Index Reg	INX													08	4	1							
Increment Stack Ptr	INS													31	4	1							
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3										
Load Stack Ptr	LDS	BE	3	3	9E	4	2	AE	6	2	BE	5	3										
Store Index Reg.	STX				DF	5	2	EF	7	2	FF	6	3										
Store Stack Ptr	STS				9F	5	2	AF	7	2	BF	6	3										
Idx Reg → Stack Ptr	TXS													35	4	1							
Stack Ptr → Idx Reg	TSX													30	4	1							

Jump and Branch Instructions

		BOOLEAN/ARITHMETIC OPERATION												COND. CODE REG.								
		RELATIVE			INDEX			EXTND			IMPLIED											
OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	5	4	3	2	1	0			
Branch Always	BRA	20	4	2																		
Branch If Carry Clear	BCC	24	4	2																		
Branch If Carry Set	BCS	25	4	2																		
Branch If = Zero	BEO	27	4	2																		
Branch If > Zero	BGE	2C	4	2																		
Branch If > Zero	BGT	2E	4	2																		
Branch If Higher	BHI	22	4	2																		
Branch If < Zero	BLE	2F	4	2																		
Branch If Lower Or Same	BLS	23	4	2																		
Branch If < Zero	BLT	2D	4	2																		
Branch If Minus	BMI	28	4	2																		
Branch If Not Equal Zero	BNE	26	4	2																		
Branch If Overflow Clear	BVC	28	4	2																		
Branch If Overflow Set	BVS	29	4	2																		
Branch If Plus	BPL	2A	4	2																		
Branch To Subroutine	BSR	8D	8	2																		
Jump	JMP				6E	4	2	7E	3	3												
Jump To Subroutine	JSR				AD	8	2	BD	9	3												
No Operation	NOP										02	2	1									
Return From Interrupt	RTI										3B	10	1									
Return From Subroutine	RTS										39	5	1									
Software Interrupt	SWI										3F	12	1									
Wait for Interrupt	WAI										3E	9	1									

Condition Code Register Manipulation Instructions

		BOOLEAN/ARITHMETIC OPERATION												COND. CODE REG.								
		IMMED			INDEX			EXTND			IMPLIED											
OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	5	4	3	2	1	0			
Clear Carry	CLC	0C	2	1																		
Clear Interrupt Mask	CLI	0E	2	1																		
Clear Overflow	CLV	0A	2	1																		
Set Carry	SEC	0D	2	1																		
Set Interrupt Mask	SEI	0F	2	1																		
Set Overflow	SEV	08	2	1																		
Accmtr A → CCR	TAP	06	2	1																		
CCR → Accmtr A	TPA	07	2	1																		

CONDITION CODE REGISTER NOTES:

- (Bit set if test is true and cleared otherwise)
- 1 (Bit V) Test: Result = 10000000?
- 2 (Bit C) Test: Result = 00000000?
- 3 (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
- 4 (Bit V) Test: Operand = 10000000 prior to execution?
- 5 (Bit V) Test: Operand = 01111111 prior to execution?
- 6 (Bit V) Test: Set equal to result of N⊕C after shift has occurred.
- 7 (Bit N) Test: Sign bit of most significant (MS) byte = 1?
- 8 (Bit V) Test: 2's complement overflow from subtraction of MS bytes?
- 9 (Bit N) Test: Result less than zero? (Bit 15 = 1)
- 10 (All) Load Condition Code Register from Stack. (See Special Operations)
- 11 (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.
- 12 (All) Set according to the contents of Accumulator A.

APPENDIX C COST

a. Microcomputer Minimum System

MPU	3000 TL
EPROM	2000 TL
RAM	2000 TL
PIA's	3000 TL
ACIA	750 TL
Miscellaneous	10000 TL

20750 TL

b. Interface Circuits

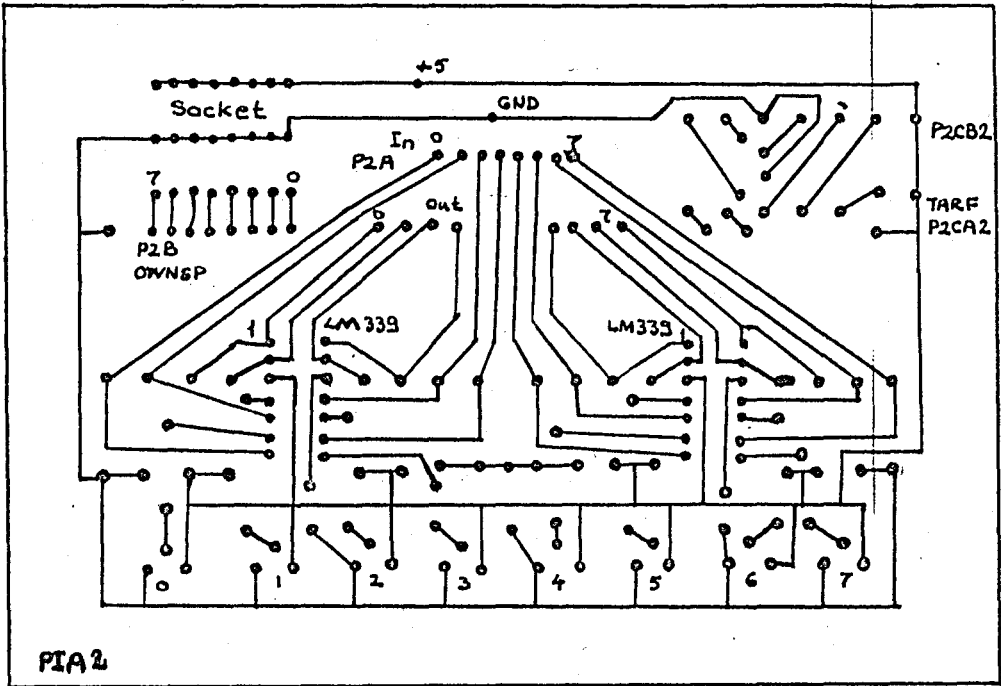
Semiconductor Components	5000 TL
Miscellaneous	3000 TL

8000 TL

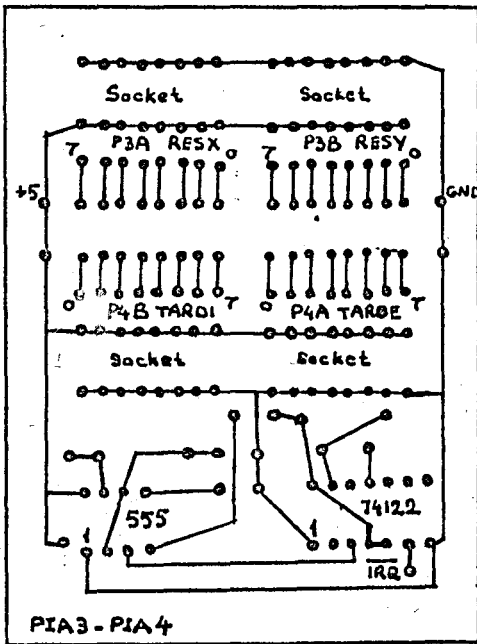
c. Engineering Cost

50000 TL

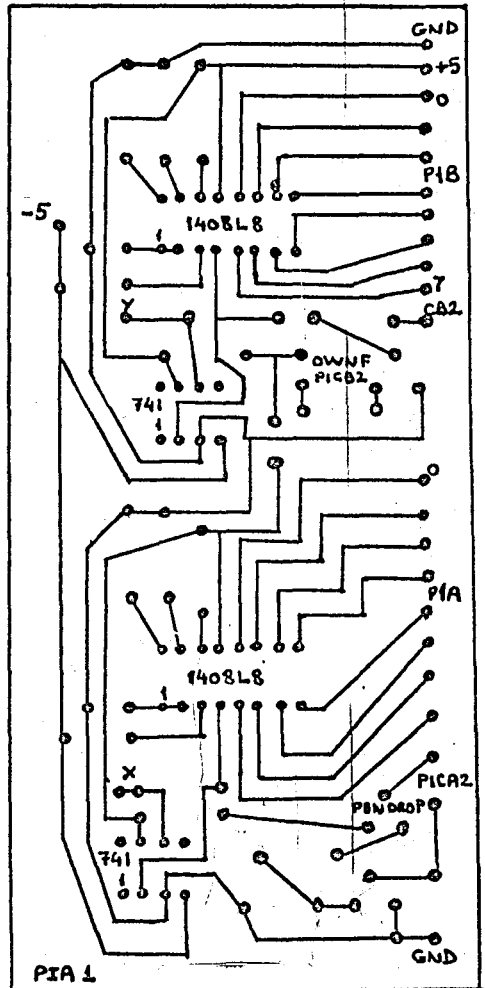
APPENDIX D PRINTED CIRCUIT BOARDS



Shaft encoder and Own Speed Input Board



Timer and Reset Coordinates, Target Information Input Board



D/A Converter Board