

AD-A081 063

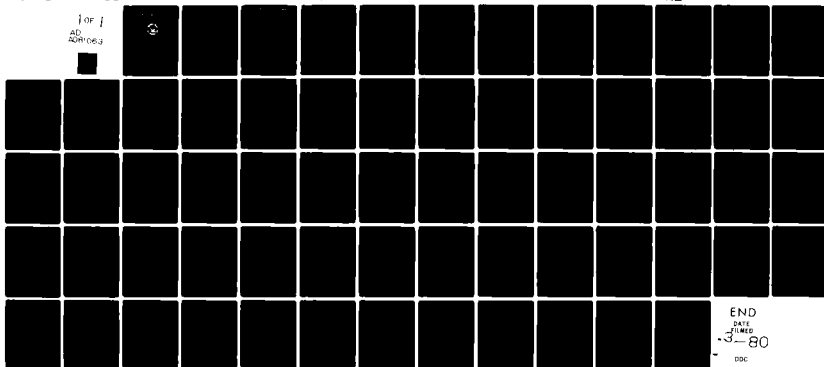
NAVAL POSTGRADUATE SCHOOL MONTEREY CA
MICROCOMPUTER DESIGN FOR ELECTRONIC EMITTER IDENTIFICATION.(U)
SEP 79 6 L BUSH

F/6 17/4

UNCLASSIFIED

NL

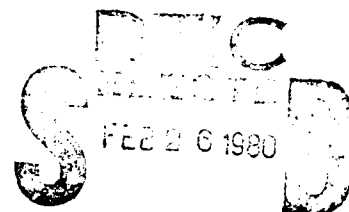
1 of 1
AD
A081063



ADA 081 063

NAVAL POSTGRADUATE SCHOOL

Monterey, California



A

DDC FILE COPY

⑨

Master's

THESIS

MICROCOMPUTER DESIGN FOR
ELECTRONIC EMITTER IDENTIFICATION

by

Gary Lew Bush

Sept 1979

Thesis Advisor:

Frank Burkhead

Approved for public release; distribution unlimited

THIS DOCUMENT IS UNCLASSIFIED
EXCEPT WHERE SHOWN
OTHERWISE BY DATA
MARKS WHICH DO NOT
CONVICT.

251 450

B

DISCLAIMER NOTICE

**THIS DOCUMENT IS BEST QUALITY
PRACTICABLE. THE COPY FURNISHED
TO DDC CONTAINED A SIGNIFICANT
NUMBER OF PAGES WHICH DO NOT
REPRODUCE LEGIBLY.**

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Microcomputer Design for Electronic Emitter Identification		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; Sept. 79
7. AUTHOR(s) Gary Lew Bush		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		12. REPORT DATE September 1979
		13. NUMBER OF PAGES 61
		15. SECURITY CLASS. (of this report) Unclassified
		16. DECLASSIFICATION/DOWNGRADING SCHEDULE
17. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
18. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
19. SUPPLEMENTARY NOTES		
20. KEY WORDS (Continue on reverse side if necessary and identify by block number) table look-up no degradation off-line		
21. ABSTRACT (Continue on reverse side if necessary and identify by block number) New technological advances in weapons and electronics has increased the fighting capabilities of small ships. The increases are seen in better and lighter offensive weapons, along with, increases in self-defensive weapons through the addition of electronic suites. The electronic warfare suites currently used on small ships are limited in size, weight,		

and response time because of the environment in which these ships are employed. It is the purpose of this paper to propose a system which will allow the currently very limited data bases to be expanded while at the same time not degrading the system. The feasibility of this approach is based upon using a microcomputer to aid in the identification of emitters off-line while leaving the current system to identify high threat emitters.

A 23
C.F.

Approved for public release; distribution unlimited

MICROCOMPUTER DESIGN FOR
ELECTRONIC EMITTER IDENTIFICATION

by

Gary L. Bush
Lieutenant, United States Navy
B.B.A., North Texas State University, 1974

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1979

Author:

Gary L. Bush

Approved by:

J. Burkhead
Thesis Advisor

A. H. Hays
Second Reader

R. T. Hays
Chairman, Computer Science Department

D. Schradt
Dean of Information and Policy Sciences

ABSTRACT

New technological advances in weapons and electronics has increased the fighting capabilities of small ships. The increases are seen in better and lighter offensive weapons, along with, increases in self-defensive weapons through the addition of electronic suites. The electronic warfare suites currently used on small ships are limited in size, weight, and response time because of the environment in which these ships are employed. It is the purpose of this paper to propose a system which will allow the currently very limited data bases to be expanded while at the same time not degrading the system. The feasibility of this approach is based upon using a microcomputer to aid in the identification of emitters off-line while leaving the current system to identify high threat emitters.

TABLE OF CONTENTS

I.	INTRODUCTION	7
II.	BACKGROUND	9
	A. SOVIET INTEREST IN ELECTRONIC WARFARE	9
	B. PASSIVE WARFARE	12
	C. SPECIAL PROBLEMS OF FAST PATROL BOATS	12
	D. THE COMPUTER IN ELECTRONIC WARFARE	13
III.	ELECTRONIC WARFARE AND EQUIPMENT REQUIREMENTS	14
	A. ELECTRONIC WARFARE REQUIREMENTS	14
	1. Performance	14
	2. Operational	15
	3. Special Applications	15
	4. Meeting of Requirements	17
	B. EQUIPMENT REQUIREMENTS	17
	1. Memory Size	17
	2. Disk Drives	18
	3. Electronic Emission Control	18
IV.	COMPUTER APPLICATION	19
	A. SEARCH PROBLEM	19
	B. COMPARISON OF SEARCH ALGORITHMS	21
	C. NEED OF SORTING	24
	D. DISK FILE STRUCTURE	25
	E. TRAINING	26
	F. OTHER APPLICATIONS	28
V.	COST AND SUPPORT	31
VI.	PROGRAM IMPROVEMENTS AND FUTURE CONSIDERATIONS	31
	A. PROGRAM IMPROVEMENTS	31

B. FUTURE CONSIDERATIONS	32
1. Operating System Modifications	32
2. Passive Plotting of Contacts	33
APPENDIX A - USER'S PROGRAM GUIDE	35
APPENDIX B - PROGRAM DOCUMENTATION	38
A. PROCEDURE REMAT	38
B. PROCEDURE SORT	43
C. PROCEDURE SEARCH	43
D. PROCEDURE TALK	55
E. MAIN PROGRAM	58
BIBLIOGRAPHY	60
DISTRIBUTION LIST	61

I. INTRODUCTION

The loss of the Israeli destroyer Eilat to the Egyptian fast patrol boat, during the October War, demonstrated the use of new technology in offensive weaponry. This demonstration paved the way for new interest in the investment of electronic warfare equipment. The interest took the form of electronic suite additions to ships that, up to now, had previously done without. It also manifested itself through improvement in the larger ships by doing away with the World War II type electronic surveillance measures (ESM) and electronic countermeasures (ECM). In their place came new, highly sophisticated, computer controlled systems.

The ECM capabilities were designed to help degrade the effectiveness of the opposing forces weapons, however, the best defense that a ship has is to avoid detection in the first place. One action to take, to meet this objective, is to turn off all active electronic devices which can give away the ship's presence. Add to this action the passive exploitation of electronic emissions from the opposing forces and you have a very effective way of avoiding detection while allowing the ship to maneuver into position for attack. Sounds good, however, there is a drawback.

The drawback is that the only way to identify electronic emissions is through their parameters. These

parameters are contained in the Electronic Order of Battle (EOB) files which today consists of approximately 14,000 known land sites with a total of 62,000 emitters; 1,200 types of commercial and combatant aircraft; 28,000 commercial and combatant surface ships; 1,100 submarines and 350 different missile systems.[1]

The current electronic warfare (EW) systems presently contain between 75 and 500 emitters in their libraries, therefore, it is quite possible that a majority of the emitters intercepted are not going to be in the system library. The commanding officers of these ships will have two options left open to them. One, remain silent in the hope that the other contact will not detect him, or two, activate active electronic devices in order to try and establish the identity of the contact. The second choice defeats the best defensive weapon the ship has and should be used only as a last resort.

II. BACKGROUND

A. SOVIET INTEREST IN ELECTRONIC WARFARE

The Soviets have for many years recognized the importance of electronic warfare in all aspects of military operations. In his book Soviet Military Strategy, Marshall V.D. Sokolovski summarized the role of electronic warfare in Soviet strategy by first identifying the dual mission of EW, namely denial and protection.

He continued: "Merely to list the uses of electronic warfare is to show how widespread are electronic countermeasures (ECM) and defense against electronic countermeasures and how serious the consequences can be. For this reason the development of electronics has now acquired the same importance as the development of missiles and nuclear weapons which cannot be used without electronic equipment." He thus also identified the electronic warfare/general warfare balance in modern warfare. The former first deputy minister of defense wrote this in the 1960's. Later in that decade the Soviets engaged in EW activity in a way which surprised many Western military observers. The operation, now a classic in EW history, was the large-scale invasion of Czechoslovakia in August 1968.

The Soviets made extensive use of ECM in the form of active jamming, together with a blanket of chaff under the cover of which they assembled and transported Soviet forces in airborne and ground units into Prague and other cities without alerting NATO forces. Soviet military doctrine dictates the use of large troop formations, a situation which makes deception difficult and thus the use of noise jamming and chaff is most appropriate. Since 1968, Warsaw Pact forces have reportedly made considerable investments in electronic warfare, with particular emphasis on training.[2]

B. PASSIVE WARFARE

Passive warfare includes a mixture of elements of EW and other passive sensors with hard kill weapon response suites. It includes the use of passive techniques, methods, equipment and training for detection (including identification, classification, and track), control (threat evaluation, weapon selection, and assignment) and engagement (target acquisition through target kill) of enemy targets. Increased interest in passive warfare was stimulated by the development of long range autonomous missiles that are capable of seeking out and destroying enemy shipping beyond the radar horizon.

Passive warfare has not gained general acceptance as a mode of naval warfare. The paucity of available data and

experience with passive techniques in operational situations hinders expansion of the theoretical concepts. Operators engage in few exercises utilizing passive techniques common to other areas of warfare but return to traditional classical methods of employing active sensors in responding to potential threats. Differences of opinion in the military worth of passive techniques and response systems among various established groups typified by the hard-kill weapon advocates and so called active EW "electron gun" enthusiasts continue to hinder the expansion and acceptance of passive warfare. Its role in respect to other modes of warfare and the needs of the total force must be resolved to counter the sophisticated threat in modern warfare.

The dependence on high-technology weapon systems which rely primarily on active sensors (e.g. radar) that uses some region of the electromagnetic spectrum for control and guidance information indicates the need for further attention of naval planners to seek the optimum mix of ESM/ECM/ECCM (electronic counter countermeasure) and weapons. No longer can fleet units continue to freely broadcast their positions to the enemy and survive. More attention to rigorous control of onboard active devices is needed along with disciplined operations which include consideration of both active and passive capabilities. The time has come to expand passive warfare and determine its role and contribution to naval warfare.[3]

C. SPECIAL PROBLEMS OF FAST PATROL BOATS

Fast patrol boats (FPBs) are not without problems, however. They are vulnerable to anti-ship, and even aircraft, missiles. A 30mm-equipped aircraft gunship can destroy a FPB in a matter of seconds. Sea skimmers are especially effective against FPBs. Payload and cost considerations do not often allow a highly effective AAA or SAM system to be placed onboard, except in the very largest FPBs such as the Soviet Nanuckla-class ships. Five-inch and larger naval guns or shore batteries have FPBs at a disadvantage if they venture too close to their intended targets or are slightly careless with their navigation. Missile carrying FPBs cannot afford to engage in duels with platforms other than their intended targets because of their limited supply of offensive weapons. The cost of FPBs is low, but only relatively speaking. To most users, the cost is not low enough to allow truly massive FPE attacks which would overwhelm enemy defenses without regard to the FPB losses. Some sort of protection is needed. Finally, a typical FPE cannot reload at sea. The first shots must be accurate and be directed at the target intended. There is essentially no "second chance." Expendng ammunition and missiles for self-defense, at spurious targets, or engaging non-hostiles requires a high degree of pre-launch confidence in target identification and location. The use of the FPB's radar does

not often give a high level of confidence that the intercepted entity is truly the target desired. The use of electronic warfare assets will largely overcome these problem areas.[4]

D. THE COMPUTER IN ELECTRONIC WARFARE

A central computer forms the nucleus of the system where the high level processing is performed and from which the executive commands would emanate. Remotely located microprocessors would be dedicated to performing the low level, routine tasks associated with real-time, hardware-related functions. ECM techniques are sufficiently flexible to allow new techniques to be implemented via software modifications rather than costly and time consuming hardware changes.[5]

The AN/SLQ-29, AN/SLQ-32, AN/SLR-21 and EW-120 series are some of the current systems using computer technology. The AN/SLQ-29 is used onboard large ships such as aircraft carriers. The AN/SLQ-32 is used onboard smaller ships such as DDG, FFG and small auxiliaries. The AN/SLR-21 and EW-120 series are used on FPEs.

III. ELECTRONIC WARFARE AND EQUIPMENT REQUIREMENTS

A. ELECTRONIC WARFARE REQUIREMENTS

1. Performance

The basis for performance requirements lies in the primary functions of the system. In the case of the AN/SIQ-32 and similar systems, the following items are of prime concern. First, surveillance which is the monitoring of the electromagnetic environment in the radio frequency bands (normally between 8 - 18 GHz) and over angular regions where hostile emitters can be expected. Secondly, the identification of an emitter is done by comparing its characteristics with those of known emitters. Third, warning which means to monitor the environment and provide a warning whenever a change occurs; for example, detection of a new emitter or change in an operational mode of an old emitter. Fourth, countermeasures are measures that are taken in the event that a new emitter has been identified as hostile and initiates countermeasures which will degrade the performance of the hostile emitter/platform. Fifth, operator/system interaction which provides controls allowing the operator to initiate or modify system operation on the basis of the situation. Finally, intersystem interaction which interfaces with other onboard systems and exchanges information as

required in order to maximize the survivability of own ship and to facilitate the deployment of other weapons.[5]

2. Operational

The basis for operational requirements lies in the following items. First, the environment which affects the ship while conducting offensive strike warfare in support of carriers or other task forces, such as underway replenishment and military convoys. Success in these strike operations, especially those operations without air support, depends upon continuous shipboard surveillance to provide target information for quick response to anti-ship missile attacks. Ideally, our ships will remain undetected while conducting extended-range surveillance with a high probability of initial detection and identification of enemy ships. Secondly, the tactics used are normally those of passive-offensive capability in which complete radar silence is maintained while utilizing rapid and reliable covert telecommunications to disseminate passive targeting data and engagement control orders.[3]

3. Special Applications

The special requirements of certain ships add additional concerns with respect to ship functions. They are as follows. First, in the case of fast patrol boats (FPBs),

their requirements for warning and acquisition receivers are not very much different from presently available electronic surveillance measurement systems. A few exceptions exist, however, in addition to the restrictions on size, weight, reliability and ease of operation. The last item is important because the level of competence available for operations of the electronic warfare system may likely be quite low. Secondly, the frequency coverage should be, as a minimum, between 8 - 18 GHz, which is the frequency range in which most threats are expected to be found presently and into the near future. Third, sensitivity, because of the need for intercept of targets and threats beyond their maximum detection range, should be better than usual crystal video receivers. In addition, relatively low mast heights on a fast patrol boat require additional sensitivity. However, increased sensitivity requires increased complexity - a highly weighted factor on FPBs. Fourth, azimuth coverage of 360 degrees would, of course, be required and elevation coverage should cover from 0 degrees (to intercept sea skimmers) to a minimum of +30 degrees. A stabilized platform with +45 degrees azimuth coverage would be much preferred. The severe pitch and roll of FPBs help create the need for stabilized antennas. Finally, a programmable threat library would be mandatory in an FPB electronic surveillance measure suite, so that it could remain small and be loaded in at the pier for the mission, the scenario expected and the operations planned.[4]

4. Meeting of Requirements

Current systems like the AN/SLQ-32 and AN/SLR-21 systems meets most of the hardware requirements but fall short when it comes to the functional and operational requirements. The shortcomings are due to the size, weight and response time constraints on small ships. However, even if the size and weight constraints could be lifted to handle larger systems, the response times of those systems would also increase, which in critical cases such as missiles, would not be acceptable. Therefore, by adding a micro-computer system off-line, there would be no degradation in the current systems. The addition of this system will, however, allow identification of emitters well beyond the limited libraries of those systems and at the same time change what was a previously data base limited system to one bound only by the decision of the operator to cease search.

B. EQUIPMENT REQUIREMENTS

1. Memory Size

The memory size of the prospective off-line micro-computer would be 48k bytes minimum. The allowance would include 4k bytes for the operating system, 12k bytes for the processing programs, and 32k bytes for library array

processing. The system should also allow a growth factor of approximately 33%, bringing the preferred memory size to 64k bytes. Most microcomputers currently have this capacity option as an off-the-shelf item and would not require any special design changes.

2. Disk Drives

The major constrainting factor of the prospective system lies in the disk drives currently available as an off-the-shelf item. The constraint takes the form of head dimensions and densities of the disk drives. The prospective system is based upon the use of an IBM 3740 style floppy disk with 77 tracks per disk, 26 sectors per track, 128 bytes per sector, and single density.

3. Electronic Emission Control

The prospective system will have to be capable of operating under electronic emission control (EMCON) conditions. It would, therefore, be necessary for the system to conform to the MIL-STD-461 for TEMPEST inspection. The requirement can be met with current systems on the market.

IV. COMPUTER APPLICATION

A. SEARCH PROBLEM

A basic requirement appearing in many data-processing problems is the need to search a mass of information for certain information associated with specific information on hand. We begin with some terminology and symbol definitions for an abstract statement of a search problem.

- F Denotes a file, here considered as a matrix.
- F[I;] Denotes a record 'I' of the file (also called an item 'I').
- F[:,J] Denotes a position 'J' of all records of the file. F[:,J] is a column vector of 'F'.
- X Denotes an argument to be used on the basis of search. 'X' will usually correspond to a part of a record and is considered a vector.
- M Denotes a mask or format vector which specifies, by its elements, which columns of 'F' are to participate in the search.

With the above terminology, a simple but common search problem may be stated in words as: Given a file 'F', a format vector 'M', and an argument 'X'. Find 'G', a submatrix of 'F' containing all those records of 'F' whose contents in the key positions match the argument.[6]

The file used here is made up of a set of records (128 bytes long) in the following format:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

1. Radio Frequency Low (RFL)
2. Radio Frequency High (RFH)
3. Pulse-repetition Frequency Low (PRFL)
4. Pulse-repetition Frequency High (PRFH)
5. Scan Period Low (SPL)
6. Scan Period High (SPH)
7. Scan Type (ST)
8. Pulse Width Low (PWL)
9. Pulse Width High (PWH)
10. Modulation Type
11. Elint Notation
12. Mode Counter (MC)
13. NATO Nickname
14. Emitter Function Code (FC)
15. CDS Number (Formally NTDS)
16. Comments

The file record is made up of file items (F[I;]) such as RFL, RFH, PRFL, etc. The file column vector (F[;J]) contains all RFLs, or all RFHs, etc., on one disk. The file is limited to 1950 records per disk, but the number of disks is not limited. Therefore, the data base is no longer limited by memory size constraints.

The argument vector 'X' is composed of radio frequency (RF), pulse-repetition frequency (PRF), scan and scan type. It is entered into the system by the operator which has read it directly from the display of the current system such as the AN/SLQ-32.

The mask vector is comprised of RFL, RFH, PRFL, PRFH, SPL, SPH and ST. The remainder of the file is used for the output information to the operator when the search is successful. As noted, the items of the argument are specific

while the items in the mask are specific only in the scan type. The other items are the upper and lower limits of the argument items and are satisfied only when the argument items lies within these limits.

B. COMPARISON OF SEARCH ALGORITHMS

The three basic search algorithms are sequential, hashing, and binary. The normal approach is to apply only one of them at a time, however, a combination of them may be used if the situation warrants. In the case of the micro-computer, a very limited memory size and the desire to maximize the number of records on the disk required keeping pointers to a minimum. It also required the file to be broken up into partitions consisting of 250 records each, which allows one partition to fit into the machine memory at any one time. Putting these records into memory vice leaving them on the disk will benefit the system by cutting down on the number of accesses to the disk while increasing the speed of the search.

In the interest of minimizing the accesses to the disk, which takes a longer time in comparison to accessing memory, a reference table was added to the disk holding the addresses of all partitions and the maximum and minimum values of the RF which is the primary search key. With this

in mind, only those partitions which satisfy the RF item in the argument will be eligible for loading into memory.

The comparisons of the search methods will be based upon the aforementioned and their particular circumstance. The common terms for the comparison are as follows:

Let A = Machine time to do one comparison.

B = Total access time of disk (including the transfer of 1 segment).

C = Time to load and process the reference table from the disk.

P = Number of partitions (max. of 8).

N = Maximum partition size (250 records).

I = Number of items in the argument.

Assumption: Only one partition needed per search.

The worst case for the sequential search method is where the record that satisfies the argument is located at the bottom of the file.

$$\begin{aligned}\text{Total Time} &= P([N * I * A] + B) + C \\ &= 1([250 * 4 * A] + B) + C \\ &= 1000A + B + C\end{aligned}$$

The hashing search method is based upon a value derived from an argument vector. In this application, the argument key is already in the form of a value which lies between two

limits. Because the hashing value is unique, it is not applicable to this situation.

The worst case for the binary search method is where the record that satisfies the argument lies either side of the mid point, or is the second record in the file or is the next to last record in the file. Mathematically it is $\log_2 N$. Its proof is to consider the binary decision tree describing the action of a binary search on n elements. All successful searches end at a node while all unsuccessful searches end at a leaf. If $2^{k-1} < n < 2^k$ (k is the number of levels in the tree) then all nodes are at the levels 1, 2, ---, k , while all leaves are at levels k and $k+1$ (note that the root node is at level 1). The number of element comparisons needed to terminate at a node on level i is i while the number of element comparisons to terminate at a leaf at level i is only $i-1$. [7] The following equation shows the total time of the search based upon that only three of the four items in the argument can be used in a search of this manner. The fourth will be searched sequentially.

Let $S = I-1$

$$\begin{aligned} \text{Total Time} &= P(S[\log_2 N] * A + B) + C + (N * A) \\ &= 1(3[8] * 1 + B) + C + (250 * A) \\ &= 24A + B + C + 250A \\ &= 274A + B + C \end{aligned}$$

Note: $(N * A)$ is the sequential search of the fourth item.

Based upon the comparison of 274 to 1000, the binary search with the modifications stated was chosen for this application.

C. NEED FOR SORTING

The use of a sequential search does not require that the file be in any particular order. However, in the case of the binary search, it is mandatory that the file be in either increasing or decreasing order because the binary search is based upon the idea that after one comparison; half of the file can be eliminated followed by a repeat of the technique on the remaining half until the search is completed. The sorting scheme used in this application was to place the file into an increasing order. There are, in this case, three fields that must be in increasing order. They are RF, PRF, and scan.

The process used was to take a partition of the file and only load the RFL, RFH, PRFL, PRFH, SPL, and SPH mask vectors into memory. During this load, each record is assigned a record number in increasing order. The next process was to divide the partition into a subset with all the same RFLs, after a bubble sort on the RFLs. Next, perform a bubble sort on all RFHs followed by dividing this vector in a subset of the subset containing all of the RFHs with the same value. The process is repeated for the

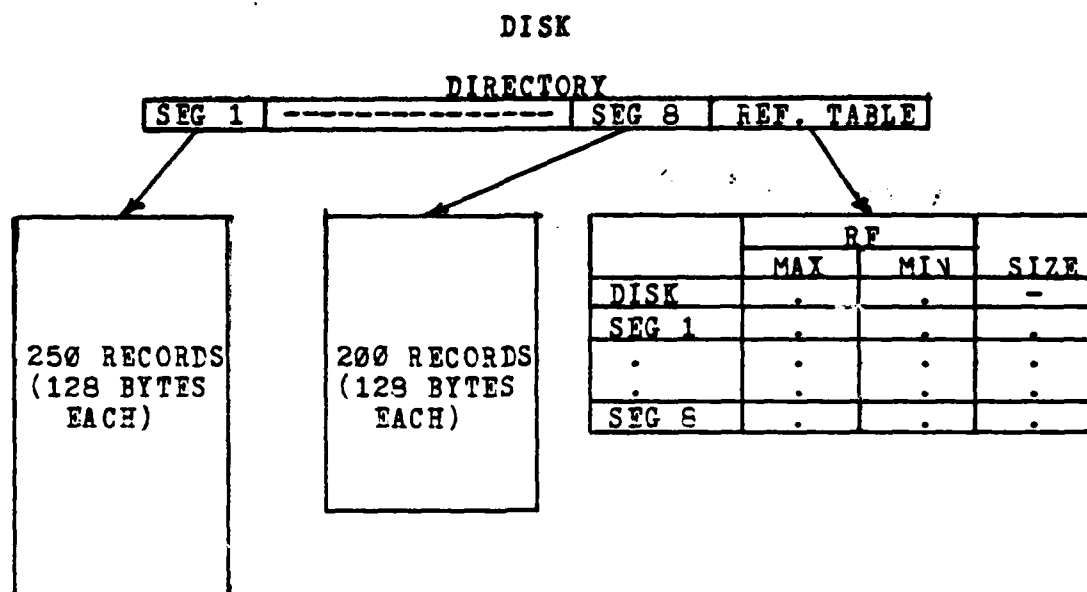
remaining items PRFL, PRFH, SPL, and SPH. Once the first pass is made on the final subset, the process is back stepped one iteration and done again. The process is continued until the entire file is sorted. For more details on the process, see the PROCEDURE SORT in APPENDIX B.

At the conclusion of the sort, the modified file will be in increasing order in all fields except the record number. It will be in disorder but in the order in which the file will have to be converted to place it into increasing order. To do this, the record number is saved in an array and used to reference each record number in the partition. The process then is to load memory with the partition in the order of the saved array, followed by rewriting the partition on the disk. Once this is done, the partition will be in increasing order and there will be no further need for modifications.

D. DISK FILE STRUCTURE

The disk contains a directory on track one. The directory contains pointers to seven segments containing 250 records each, one segment with 200 records, and a reference table. The reference table is made up of the maximum and minimum RF values of each segment and the minimum and maximum RF value for the disk. It is this table that points to what segment or segments will be loaded into memory.

The logical arrangement of the disk is as follows:



The disk is not supplied in the proper format because it would mean that the support facility would have to access all of the different kinds of equipment that are in the fleet. It is not necessary for the support facility to have all of the equipment involved if it can produce the data in a standard format and on standard media. It can later be modified by the user's software to conform to any special needs of the user.

The reformatting program PROCEDURE REMAT can be seen in the APPENDIX B. The program reads 512 bytes of source data of which only 128 bytes (due to the dissimilarity between operating systems) is written into the segment at any one

time. The order of writing is first into segment one, then when 250 records have been written, it writes segment two and so on until all records are written onto the disk.

The restrictions of using track one thru seventy-six, the reading of 512 byte blocks, and limiting the arrays to a maximum of 16k words are imposed by the UCSD PASCAL system.[8] It is these restrictions that prompted the breaking up of the source file into eight segments. Another restriction was the format of the disk which makes best use of 128 byte records.

The support facility's record size is compatible but of different arrangement than what would be necessary for efficiency in this application and was therefore changed. The first two diagrams represent how the data is received and the third diagram is how it is reformatted into an efficient form for this program.

IDENTIFICATION RECORD

1	2	3	4	5	6
---	---	---	---	---	---

1. Elint Notation	0-4
2. Mode Counter	7
3. NATO Nickname	9-20
4. Emitter Function Code	22-23
5. Four Digit CDS Number	26-29
6. Comments	37-60

PARAMETER RECORD

1	2	3	4	5	6	7	8	9	10	11	12	13
---	---	---	---	---	---	---	---	---	----	----	----	----

1.	Elint Notation	0-4
2.	End Mode	5
3.	Mode Counter	6-7
4.	RF Low	9-13
5.	RF High	15-19
6.	PRF Low	21-25
7.	PRF High	27-31
8.	Pulse Width Low	33-36
9.	Pulse Width High	38-41
10.	Modulation Type	44
11.	Scan Type	46-50
12.	Scan Period Low	52-55
13.	Scan Period High	57-60

REFORMATTED RECORD DESIGN

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

1.	RFL	0-4
2.	RPH	6-10
3.	PRFL	12-16
4.	PPFH	18-22
5.	SPL	24-27
6.	SPH	29-32
7.	ST	35-39
8.	Pulse Width Low	41-44
9.	Pulse Width High	46-49
10.	Modulation Type	51
11.	Elint Notation	64-68
12.	Mode Counter	70
13.	NATO Nickname	72-84
14.	Emitter Function Code	85-86
15.	CIS Number	88-91
16.	Comments	92-116

E. TRAINING

The capabilities required of the operator is that he be able to read and follow some simple instructions. He will also be required to enter information into the system from the display unit of the current detection equipment, such as an AN/SLQ-32. No other special skills will be required of

him; therefore, he will not have to have any training off of the ship.

F. OTHER APPLICATIONS

The proposed system should be used as a dedicated system to help identify electronic emitters while at sea. However, when in port there are areas where the microcomputer could be used to lighten the burden of administration. Some of these areas are in the reports and reporting fields. Godley [10], while at the Naval Postgraduate School, commented on how a microcomputer could be used in recurring operational reports and how this would benefit the Navy. Text processors are also commercially available for microcomputers and they too would help in formatting letters and messages. It then is up to the respective commanding officer to determine just how much a microcomputer can help his command.

V. COST AND SUPPORT

The basic system is comprized of a terminal (consisting of a video screen and keyboard), a microcomputer of the 8080 or 280 design with 64k of memory, and dual 8" floppy disk drives. The approximate cost of the system is as follows:

Terminal	\$1000
Microcomputer with drives	<u>\$4000</u>
TOTAL	\$5000

As an option, a printer can be added to the basic system for input use. The system can be purchased by the type commander or, according to a new regulation, by the commanding officers of their respective ships because the system's cost is less than \$10,000.

The data base support would be from Naval Electronic Evaluation Office, Damneck, Virginia. Contact should be made with them prior to any purchase of any equipment to verify their requirements for continued support.

VI. PROGRAM IMPROVEMENTS AND FUTURE CONSIDERATIONS

A. PROGRAM IMPROVEMENTS

The two currently available operating systems CPM and UCSD PASCAL were the driving forces behind their selection for this project. However, with new operating systems continually being developed the reformatting software program was built with the idea of flexibility. It can be modified to read any sector on the disk and transcribe it into the correct format needed for processing under the available programs. Studying the reformatting program closely will reveal that if the operating system is using a directory listing on any track (except track one) the listing can be pulled off. Then by developing an equation to decode this information all sectors can be pulled off in the correct order.

The only thing to check is that the initial data stored on the disk, supplied by the support facility, is the same as the data described earlier in this paper. Given that it is, the only constraining factor is the hardware. The hardware constraint is that the disk be of the type IBM 3740 style floppy disk with 77 tracks per disk, 26 sectors per track, 128 bytes per sector, and of single density.

The improvement of this program will come about through expansions due to decoding of new operating systems disk directories. It will also be necessary thru interactive communications with the low skilled operator to establish who supplied the disk, thereby establishing which equation to use in reformatting the input data. The process in which the reformatting is done should remain transparent to the operator. He should only have to respond to questions presented to him on the CRT and load the disk as directed.

B. FUTURE CONSIDERATIONS

1. Operating System Modifications

The normal typist will type approximately 30 to 60 (five letter) words per minute. Using the 60 word per minute as a norm, the time interval between characters of input then is two tenths of a second. Allowing for a 50% safety factor for keybounce, settling out of transient signals and various other static factors, leaves one tenth of a second that could be used for processing. Agreed that this does not sound like much time in the normal sense, to a computer with instruction times between 4 - 10 micro seconds this is 10,000 - 25,000 instructions that could have been done without any delays in the inputs from the keyboard.

With this in mind, a modification to the operating system would increase the response time to the user without any degradation to the system. In addition, if a queueing scheme, which would allow stacking of request for processing, was included and implemented with a priority system, the over all system performance could be improved.

The consideration of the operating system modification would have to be based upon sound computer science studies. The field is open and presents a challenge to any student who is interested in applying this concept to a table look-up identification scheme.

2. Passive Plotting of Contacts

The biggest problem with plotting a contact through passive intercept is with the intermittent short signals. These signals do not allow the intercepting platform to get a cut on the signal and proceed to a distant point and again get another cut, thus allowing the position to be plotted.

Couple the system described in this paper and one that is currently being done on milliwave transmissions by Mark Schneider and Mike Chase at the Naval Postgraduate School and these intermittent signals can be plotted. The way that it will work is that digital signals can be transmitted over these milliwaves in such a manner that unintentional dis-

closures of own ship's position is fairly unlikely. It will require that two ships which have the equipment are within sight of one another and each ship gets at least one cut on the intermittent signal, followed by a transmission over the milliwave system to establish the distance between the two ships through the same type of process now being used in most radars (a send and receive signal vs time). With the bearing cuts of each ship plus their distances apart at the time of intercept, a plot of the contact can be made.

APPENDIX A

USER'S PROGRAM GUIDE

The program was implemented on the ALTOS microcomputer system with a DATAMEDIA ELITE 2500 terminal. The booting and loading instructions for this system will be used for examples. However, if a different system is being used, the booting instructions may vary, so the operator's manual must be checked. Once the system is booted and the screen writes its welcome message, all other instructions will be the same.

Example 1: Booting

- Step 1. Turn power on (it is located on the back of the microcomputer and on the right hand side of the CRT terminal).
- Step 2. Place the disk labeled 'SYSTEM' into the right hand side of the drive with the label faced to the operator and down.
- Step 3. Depress the key labeled D/2 if the full duplex lamp (located on front of the CRT terminal) is off.
- Step 4. If the symbols '%*' are not on the screen depress the rest button located on back of the microcomputer.
- Step 5. If the symbols '%*' are on the screen depress the key labeled LOCK and verify that it remains in the down position.
- Step 6. Depress the key labeled 'U' followed by depressing the key labeled RETURN.

The system should be booted at this time. If the screen does not display the welcoming message repeat the steps two thru six. The welcoming message is as follows:

```
Command: E(dit, R(un, F(ile, C(ompile, H(alt.  
WELCOME USER, TO  
U.C.S.D. PASCAL SYSTEM 'I.xx
```

The next phase to complete is the loading instructions and they will be the same for all systems.

Example 2: Loading

- Step 1. Depress the key labeled 'F'. The screen will display 'Filer: G(et, S(ave, W(hat, Q(uit'.
- Step 2. Depress the key labeled 'G'. The screen will display 'Get what file?'.
- Step 3. Depress the keys labeled 'I', 'D' and RETURN. The screen will display 'Text and code file loaded.'
- Step 4. Depress the key labeled 'Q'. The screen will display 'Command: E(dit, R(un, F(ile, C(ompile'.
- Step 5. Depress the key labeled 'R'. The screen will display 'WELCOME TO THE ELECTRONIC EMITTER IDENTIFIER'.

The booting and loading is now completed. The micro-computer will converse with the user through some simple instructions. In response to these instructions the user will be asked to make entries. These entries are of the form of answers to questions. If the user is asked for (Y/N) this means for him to depress the key labeled 'Y' for yes or the

key labeled 'N' for no. In any other response, the response will be followed by depressing the key labeled RETURN. The statements requesting this type of response will have <rtn> attached to them, such as PRF(xxxxx) <rtn>. In response to entries requiring number inputs such as PRF(xxxxx) <rtn>, the x's represent numeric numbers and the entry should contain only numbers. In the case of SCAN TYPE (AAAA), the response must be alpha characters and in the same amount that is enclosed in the parenthesis (). If the scan type display on the electronic detection equipment has less than that shown in the parenthesis, spaces must be added to the leading character, such as CON the entry would be <space>CON. The final case, is where the user is asked if he wants to change any parameters. An example of this type follows:

YOUR ENTRIES ARE:

PRF	RF	SCAN	SCAN TYPE
00005	01035	00020	CON

DO YOU WANT TO CHANGE ANY? (Y/N) Y

WHICH ONE DO YOU WANT TO CHANGE (PRF,RF,SCAN,SCAN TYPE) <rtn>
SCAN TYPE

ENTER SCAN TYPE (AAAA) <rtn> SWEP

YOUR ENTRIES ARE:

PRF	RF	SCAN	SCAN TYPE
00005	01035	00020	SWEP

DO YOU WANT TO CHANGE ANY? (Y/N) N

APPENDIX B PROGRAM DOCUMENTATION

A. PROCEDURE REMAT

```

PROCEDURE REMAT;
VAR BLOCKNR, LENGTH, UNITNR: INTEGER;
    TRACK, SETOFF, STCONE: INTEGER;
    SEGCNT, CPMSEC, CNT, LINE: INTEGER;
    TABLE, DFILE: TEXT;
    SEGMEX: STRING[11];
    CPMARAY: PACKED ARRAY[0..511] OF CHAR;

```

```

FUNCTION CPM(SECTOR: INTEGER): INTEGER;
BEGIN
    SECTOR := SECTOR + 6;
    IF SECTOR < 26 THEN CPM := SECTOR
    ELSE
        BEGIN
            IF SECTOR < 27 THEN
                BEGIN
                    SECTOR := SECTOR MOD 26;
                    IF SECTOR = 0 THEN SECTOR := 1;
                    CPM := SECTOR;
                END
            ELSE
                CPM := 0;
            END;
        END;
END;

```

```

FUNCTION OFF(CPMSEC, SECTOR: INTEGER): INTEGER;
VAR OFFSET, STPSEC: INTEGER;
BEGIN
    OFFSET := 0;
    IF SECTOR = 0 THEN STPSEC := 24
    ELSE
        STPSEC := SECTOR - 2;
        WHILE SECTOR < CPMSEC DO
            BEGIN
                OFFSET := OFFSET + 1;
                SECTOR := SECTOR + 1;
                IF SECTOR >= 26 THEN SECTOR := SECTOR MOD 26;
                IF (SECTOR = STPSEC) AND (SECTOR < CPMSEC)
                    THEN SECTOR := STPSEC - 1;
                IF OFFSET > 3 THEN
                    BEGIN
                        OFFSET := 0;
                        BLOCKNR := BLOCKNR + 1;
                    END;
                END;
            END;
        END;
    END;
END;

```

The system should be booted at this time. If the screen does not display the welcoming message repeat the steps two thru six. The welcoming message is as follows:

```
Command: E(dit, R(un, F(ile, C(ompile, H(alt.  
WELCOME USER. TO  
U.C.S.D. PASCAL SYSTEM 'I.xx
```

The next phase to complete is the loading instructions and they will be the same for all systems.

Example 2: Loading

- Step 1. Depress the key labeled 'F'. The screen will display 'Filer: G(et, S(ave, W(hat, Q(uit'.
- Step 2. Depress the key labeled 'G'. The screen will display 'Get what file?'.
- Step 3. Depress the keys labeled 'I', 'D' and RETURN. The screen will display 'Text and code file loaded.'
- Step 4. Depress the key labeled 'Q'. The screen will display 'Command: E(dit, R(un, F(ile, C(ompile'.
- Step 5. Depress the key labeled 'R'. The screen will display 'WELCOME TO THE ELECTRONIC EMITTER IDENTIFIER'.

The booting and loading is now completed. The micro-computer will converse with the user through some simple instructions. In response to these instructions the user will be asked to make entries. These entries are of the form of answers to questions. If the user is asked for (Y/N) this means for him to depress the key labeled 'Y' for yes or the

key labeled 'N' for no. In any other response, the response will be followed by depressing the key labeled RETURN. The statements requesting this type of response will have <rtn> attached to them, such as PRF(xxxxx) <rtn>. In response to entries requiring number inputs such as PRF(xxxxx) <rtn>, the x's represent numeric numbers and the entry should contain only numbers. In the case of SCAN TYPE (AAAA), the response must be alpha characters and in the same amount that is enclosed in the parenthesis (). If the scan type display on the electronic detection equipment has less than that shown in the parenthesis, spaces must be added to the leading character, such as CON the entry would be <space>CON. The final case, is where the user is asked if he wants to change any parameters. An example of this type follows:

YOUR ENTRIES ARE:

PRF	RF	SCAN	SCAN TYPE
00005	01035	00020	CON

DO YOU WANT TO CHANGE ANY? (Y/N) Y

WHICH ONE DO YOU WANT TO CHANGE (PRF,RF,SCAN,SCAN TYPE) <rtn>
SCAN TYPE

ENTER SCAN TYPE (AAAA) <rtn> SWEP

YOUR ENTRIES ARE:

PRF	RF	SCAN	SCAN TYPE
00005	01035	00020	SWEP

DO YOU WANT TO CHANGE ANY? (Y/N) N

APPENDIX B PROGRAM DOCUMENTATION

A. PROCEDURE REMAT

```

PROCEDURE REMAT;
VAR BLOCKNR, LENGTH, UNITNR: INTEGER;
    TRACK, SETOFF, STCONF: INTEGER;
    SEGCNT, CPMSEC, CNT, LINE: INTEGER;
    TABLE, DFILE: TEXT;
    SEGMEY: STRING[11];
    CPMARRAY: PACKED ARRAY[0..511] OF CHAR;

FUNCTION CPM(SECTOR: INTEGER): INTEGER;
BEGIN
    SECTOR := SECTOR + 6;
    IF SECTOR < 26 THEN CPM := SECTOR
    ELSE
        BEGIN
            IF SECTOR < 27 THEN
                BEGIN
                    SECTOR := SECTOR MOD 26;
                    IF SECTOR = 0 THEN SECTOR := 1;
                    CPM := SECTOR;
                END
            ELSE
                CPM := 0;
            END;
        END;
END;

FUNCTION OFF(CPMSEC, SECTOR: INTEGER): INTEGER;
VAR OFFSET, STPSEC: INTEGER;
BEGIN
    OFFSET := 0;
    IF SECTOR = 0 THEN STPSEC := 24
    ELSE
        STPSEC := SECTOR - 2;
        WHILE SECTOR <> CPMSEC DO
            BEGIN
                OFFSET := OFFSET + 1;
                SECTOR := SECTOR + 1;
                IF SECTOR >= 26 THEN SECTOR := SECTOR MOD 26;
                IF (SECTOR = STPSEC) AND (SECTOR <> CPMSEC)
                    THEN SECTOR := STPSEC - 1;
                IF OFFSET > 3 THEN
                    BEGIN
                        OFFSET := 0;
                        BLOCKNR := BLOCKNR + 1;
                    END;
            END;
        END;
END;

```

```

    ODD := OFFSET;
END;

FUNCTION EVEN (CPMSEC, SECTOR: INTEGER): INTEGER;
VAR OFFSET, STRSEC: INTEGER;
BEGIN
    OFFSET := 2;
    IF SECTOR = 0 THEN STRSEC := 24
    ELSE
        STRSEC := SECTOR - 2;
    WHILE SECTOR <> CPMSEC DO
        BEGIN
            OFFSET := OFFSET + 1;
            SECTOR := SECTOR + 1;
            IF SECTOR >= 26 THEN SECTOR := SECTOR MOD 26;
            IF (SECTOR = STRSEC) AND (SECTOR <> CPMSEC)
            THEN SECTOR := STRSEC - 1;
            IF OFFSET > 3 THEN
                BEGIN
                    OFFSET := 0;
                    BLOCKNR := BLOCKNR - 1;
                END;
            END;
        END;
    EVEN := OFFSET;
END;

PROCEDURE PAGE1;
BEGIN
    CLRPAGE;
    WRITELN('PLACE DISK LABELED SUPPORT FACILITY');
    WRITELN('INTO THE LEFT SIDE DISK DRIVE WITH THE');
    WRITELN('LABEL FACING YOU AND DOWN');
    WRITELN;
    WRITELN('REMOVE THE DISK LABELED SYSTEM FROM THE');
    WRITELN('RIGHT SIDE DISK DRIVE AND SAVE FOR REUSE. ');
    WRITELN;
    WRITELN('PLACE A BLANK DISK INTO THE RIGHT SIDE DISK');
    WRITELN('DRIVE WITH THE LABEL FACING YOU AND DOWN');
    WRITELN;
    WRITELN;
    WRITELN('HAVE YOU COMPLETED THIS AND ARE BOTH DISK');
    WRITELN('DRIVE DOORS CLOSED? (Y/N)');
    READ(C);
    WHILE C <> 'Y' DO
        READ(C);
    END;
    CLRPAGE;
    WRITELN('WORKING');
END;

PROCEDURE PAGE2;
BEGIN
    CLRPAGE;
    WRITELN('COMPLETED');
    WRITELN;

```

```

WRITELN('REMOVE DISK FROM LEFT SIDE DRIVE AND STORE. ');
WRITELN;
WRITELN('REMOVE DISK FROM RIGHT SIDE DRIVE AND PLACE ');
WRITELN('INTO LEFT SIDE DISK DRIVE. ');
WRITELN;
WRITELN('PLACE DISK LABELED SYSTEM INTO THE RIGHT ');
WRITELN('SIDE DISK DRIVE WITH LABEL FACING YOU AND DOWN. ');
WRITELN;
WRITELN;
WRITELN('HAVE YOU COMPLETED THIS AND ARE BOTH DISK ');
WRITELN('DRIVE DOORS CLOSED? (Y/N) ');
READ(C);
WHILE C <> 'Y' DO
  READ(C);
  BLKPAGE;
END;

```

```

PROCEDURE BUILD(CNT:INTEGER);
VAR I,J: INTEGER;

```

```

  PROCEDURE RT(X,Y:INTEGER)
  BEGIN
    WHILE X < 24 DO
      BEGIN
        WRITE(DFILE, CPMARAY[Y];
        Y := Y + 1;
        X := X + 1;
      END;
    END;
  END;

```

```

  PROCEDURE SP(X,Y: INTEGER);
  BEGIN
    WHILE X < 10 DO
      BEGIN
        WRITE(DFILE, CPMARAY[Y];
        Y := Y + 1;
        X := X + 1;
      END;
    END;
  END;

```

```

  PROCEDURE ST(X,Y: INTEGER);
  BEGIN
    WHILE X < 6 DO
      BEGIN
        WRITE(DFILE, CPMARAY[Y];
        Y := Y + 1;
        X := X + 1;
      END;
    END;
  END;

```

```

  PROCEDURE PW(X,Y: INTEGER);
  BEGIN
    WHILE Y < 62 DO
      BEGIN

```

```

        WRITE(DFILE, CPMARAY[Y];
        Y := Y + 1;
        X := X + 1;
    END;
END;

BEGIN (* BUILD *)
    I := 0;
    IF LINE < 250 THEN
        BEGIN
            J := CNT + 73;
            RF(I,J);
            J := CNT + 116;
            SP(I,J);
            J := CNT + 119;
            ST(I,J);
            J := CNT + 97;
            SP(I,J);
            WRITE(DFILE, CPMARAY[185]);
            WHILE I < 11 DO
                BEGIN
                    WRITE(DFILE, ' ');
                    I := I + 1;
                END
                Writeln(DFILE);
                I := 0;
                J := CNT;
                PW(I,J);
                Writeln(DFILE);
                LINE := LINE + 1;
            END
        ELSE
            BEGIN
                CLOSE(DFILE, LOCK);
                Writeln(TABLE, LINE);
                SEGCNT := SEGCNT + 1;
                CASE SEGCNT OF
                    2: SEGMEY := 'SEG2.TEXT';
                    3: SEGMEY := 'SEG3.TEXT';
                    4: SEGMEY := 'SEG4.TEXT';
                    5: SEGMEY := 'SEG5.TEXT';
                    6: SEGMEY := 'SEG6.TEXT';
                    7: SEGMEY := 'SEG7.TEXT';
                    8: SEGMEY := 'SEG8.TEXT';
                END;
                REWRITE(DFILE, SEGMEY);
                LINE := 0;
                BUILD(CNT);
            END;
        END; (* END BUILD *)

    BEGIN (* REFORMAT *)
        REWRITE(DFILE, 'SEG1.TEXT');
        REWRITE(TABLE, 'REFTAB.TEXT');
    END;

```

```

CPMSEC := 19;
TRACK := 2;
UNITNR := 5;
LENGTH := 512;
LINE := 0;
SEGCNT := 1;
WHILE (NOT EOF) AND (TRACK < 76) DO
  BEGIN
    BLOCKNR := (TRACK - 1) * 65 DIV 10;
    SECONE := 0 + (TRACK - 1) * 6;
    IF SECONE >= 26 THEN SECONE := SECONE MOD 26;
    IF ((TRACK - 1) * 65 DIV 10) = (((TRACK - 1) * 65) + 5) DIV 10
      THEN SETOFF := ODD(CPMSEC, SECONE)
      ELSE
        SETOFF := EVEN(CPMSEC, SECONE);
    UNITREAD(UNITNR, CPMARAY, LENGTH, BLOCKNR);
    CNT := SETOFF * 128;
    BUILD(CNT);
    CPMSEC := CPM(CPMSEC);
    IF CPMSEC = 0 THEN TRACK := TRACK + 1;
  END;
  CLOSE(TFILE, LOCK);
  WRITELN(TABLE, LINE);
  CLOSE(TABLE, LOCK);
END;

```

B. PROCEDURE SORT

PROCEDURE SORT;

CONST RFL = 0;
 FFH = 1;
 PRFL = 2;
 PRFH = 3;
 SPL = 4;
 SPH = 5;
 RECNR = 6;

VAR TABLE, DFILE : TEXT;
 SEGMEY: STRING[14];
 PTRTAB, SEGCNT, SIZE: INTEGER;
 POINT, TP, BP: INTEGER;
 RTAB: PACKED ARRAY[0..8,0..2] OF INTEGER;
 ITAB: PACKED ARRAY[0..249,0..6] OF INTEGER;
 CSEG: PACKED ARRAY[0..249,0..127] OF CHAR;
 DISKL, DISKH: INTEGER;

PROCEDURE INFEAD(BP:INTEGER);
 VAR I,J,K,L,M,N, PTR: INTEGER;
 BEGIN

 PTR := 0;
 WHILE PTR <= BP DO
 BEGIN
 READ(TFILE,I,J,K,L,M,N);
 READLN(DFILE);
 DTAB[PTR,RFL] := I;
 DTAB[PTR,FFH] := J;
 DTAB[PTR,PRFL] := K;
 DTAB[PTR,PRFH] := L;
 DTAB[PTR,SPL] := M;
 DTAB[PTR,SPH] := N;
 DTAB[PTR,RECNR] := PTR;
 PTR := PTR + 1;
 ENDLN(DFILE);
 END;

END;

PROCEDURE BUBBLE(TP,PARA:INTEGER);

VAR I,TEMP:INTEGER;

BEGIN

 WHILE TP < POINT DO

 BEGIN

 WHILE (DTAB[TP,PARA] <= DTAB[TP + 1,PARA])

 AND (TP < POINT) DO TP := TP + 1;

 IF TP < POINT THEN

 BEGIN

 I := 0;

 WHILE I <= RECNR DO

 BEGIN

 TEMP := DTAB[TP,I];

```

        DTAB[TP,I] := DTAB[TP + 1, I];
        DTAB[TP - 1,I] := TEMP;
        I := I + 1;
    END;
END;
IF (TP < PCINT) AND (TP > 0) THEN
    TP := TP - 1;
END;
END;

```

```

PROCEDURE UPDATF;
VAR I,J,K: INTEGER;
    A: CHAR;
BEGIN
    I := 0;
    WHILE I <= BP TO
    BEGIN
        RESET('DFILE');
        J := DTAB[I,RECNR];
        J := J * 2;
        K := 0;
        WHILE K < J DO
            BEGIN
                READLN('DFILE');
                K := K + 1;
            END;
        J := J - 1;
        WHILE J < 127 DO
            BEGIN
                READ('DFILE',A);
                CSEG[I,J] := A;
                J := J - 1;
            END;
        I := I + 1;
        READLN('DFILE');
    END;
    RESET('DFILE');
    I := 0;
    J := 0;
    WHILE I <= BP TO
    BEGIN
        WHILE J < 63 DO
            BEGIN
                WRITE('DFILE',CSEG[I,J]);
                J := J + 1;
            END;
            WRITELN('DFILE');
            WHILE J < 127 DO
                BEGIN
                    WRITE('DFILE',CSEG[I,J]);
                    J := J + 1;
                END;
            WRITELN('DFILE');
            J := 0;
        I := I + 1;
    END;

```

```

    I := I + 1;
END;
CLOSE(DFILE.LOCK);
END;

PROCEDURE GROUP(PARA:INTEGER);
VAR I: INTEGER;
BEGIN
    I := TP;
    WHILE (DTAB[I, PARA] = DTAB[I + 1, PARA])
    AND (I <= EP) DO I := I + 1;
    POINT := I;
END;

```

```

PROCEDURE TAB:
BEGIN
    RTAB[PTRTAB, 0] := DTAB[0, RFL];
    RTAB[PTRTAB, 1] := DTAB[EP, REF];
    RTAB[PTRTAB, 2] := SIZE;
    IF DISKL = DTAB[0, RFL] THEN
        DISKL := DTAB[0, RFL];
    IF DISKE < DTAB[EP, REF] THEN
        DISKE := DTAB[EP, REF];
    PTRTAB := PTRTAB + 1;
END;

```

```

PROCEDURE UPTAB:
VAR I, J: INTEGER;
BEGIN
    RESET(TABLE);
    WRITE(TABLE, DISKL, ' ', DISKE, ' ');
    WRITELN(TABLE);
    J := 0;
    IF (PTRTAB > 0) AND (PTRTAB < 9) THEN
    BEGIN
        WHILE J < PTRTAB DO
        BEGIN
            I := 0;
            WHILE I < 3 DO
            BEGIN
                WRITE(TABLE, RTAB[J, I], ' ');
                I := I + 1;
            END;
            J := J + 1;
            WRITELN(TABLE);
        END;
    END;
    WRITELN(TABLE, '000 000 999');
    CLOSE(TABLE.LOCK);
END;

```

```

PROCEDURE ORDER:
BEGIN
    WHILE (TP < EP) AND (TP < POINT) DO

```

```

BEGIN
  GROUP(RFL);
  IF TP < POINT THEN
    BEGIN
      BUBBLE(TP,RFH);
      GROUP(RFH);
      IF TP < POINT THEN
        BEGIN
          BUBBLE(TP,PRFL);
          GROUP(PRFL);
          IF TP < POINT THEN
            BEGIN
              BUBBLE(TP,PRFH);
              GROUP(PRFH);
              IF TP < POINT THEN
                BEGIN
                  BUBBLE(TP,SPL);
                  GROUP(SPL);
                  IF TP < POINT THEN
                    BEGIN
                      BUBBLE(TP,SPH);
                      TP := POINT;
                    END
                  ELSE
                    TP := POINT;
                END
              ELSE
                TP := POINT;
            END
          ELSE
            TP := POINT;
          END
        ELSE
          TP := POINT;
        END
      ELSE
        TP := POINT;
      END
    END
  END;

BEGIN (* SORT *)
  RESET(TABLE,'REFTAB.TEXT');
  READ(TABLE,SIZE);
  READLN TABLE;
  SEGCNT := 0;
  IF (SIZE > 0) AND (SIZE < 250) THEN
    SEGCNT := SEGCNT + 1;
  ELSE
    BEGIN
      WRITELN('UNABLE TO PROCESS FILE. CHECK DISK IN');
      WRITELN('LEFT SIDE DRIVE AND VERIFY THAT IT IS');
      WRITELN('NOT LABELED SUPPORT FACILITY. ');
      WRITELN;
      WRITELN('PLACE THE DATA DISK IN THE LEFT SIDE DISK');
    END
  END

```

```

WRITELN('DRIVE AND RELOAD. ');
WRITELN;
WRITELN('ARE YOU READY TO RELOAD? (Y/N)');
READ(C);
WHILE C <> 'Y' DO
  READ(C);
EXIT(SORT);
END;
BLKPAGE;
WRITELN('WORKING');
PTRTAB := 0;
DISKL := 32726;
DISKH := 0;
WHILE SEGCNT < 9 DO
BEGIN
  CASE SEGCNT OF
    1: SEGMET := 'SEG1.TEXT';
    2: SEGMET := 'SEG2.TEXT';
    3: SEGMET := 'SEG3.TEXT';
    4: SEGMET := 'SEG4.TEXT';
    5: SEGMET := 'SEG5.TEXT';
    6: SEGMET := 'SEG6.TEXT';
    7: SEGMET := 'SEG7.TEXT';
    8: SEGMET := 'SEG8.TEXT';
  END;
  RESET(DFILE, SEGMET);
  BP := SIZE - 1;
  INHEAD(BP);
  TP := 0;
  POINT := BP;
  BUBBLE(TP, REL);
  WHILE TP < BP DO
  BEGIN
    ORDER;
    TP := TP + 1;
    POINT := BP;
  END;
  TAB;
  UPDATE;
  READ(TABLE.SIZE);
  READLN(TABLE);
  IF (SIZE > 0) AND (SIZE < 9) THEN
    SEGCNT := SEGCNT + 1;
  ELSE
    SEGCNT := SEGCNT + 9;
  END;
  UPSTAB;
END; (* SORT *)

```

C. PROCEDURE SEARCH

PROCEDURE SEARCH;

TYPE ITEM = RECORD

REF: INTEGER;
PFF: INTEGER;
PRFL: INTEGER;
PRFH: INTEGER;
SPL: INTEGER;
SPH: INTEGER;
ST: STRING[5]
END;

VAR DFILE, TABLE: TEXT;
DTAB: PACKED ARRAY[0..249] OF ITEM;
FOUND, TP, BP, POINT: INTEGER;
RF, PRF, SCAN, SIZE: INTEGER;
SCANTYPE: STRING[5];
KEEP: BOOLEAN;
SEGMEY: STRING[14];

PROCEDURE GROUPREF(I:INTEGER);

BEGIN
IF I > 0 THEN
BEGIN
WHILE (DTAB[I].REF = DTAB[I-1].REF) AND (I > TP) DO
I := I - 1;
TP := I;
END
ELSE
TP := 0;
IF I < 249 THEN
BEGIN
WHILE (DTAB[I].REF = DTAB[I+1].REF) AND (I < BP) DO
I := I + 1;
BP := I;
END
ELSE
BP := 249;
END;

PROCEDURE GROUPPRF(I:INTEGER);

BEGIN
IF I > 0 THEN
BEGIN
WHILE (DTAB[I].PRFL = DTAB[I-1].PRFL) AND (I > TP) DO
I := I - 1;
TP := I;
END
ELSE
TP := 0;
IF I < 249 THEN

```

BEGIN
  WHILE DTAB[I].PRFL = DTAB[I-1].PRFL AND (I < EP) DO
    I := I + 1;
  EP := I;
  END
ELSE
  EP := 249;
END;

```

```

PROCEDURE GROUPSCAN(I:INTEGER);
BEGIN
  IF I > 0 THEN
    BEGIN
      WHILE(DTAB[I].SPL = DTAB[I-1].SPL) AND (I > TP) DO
        I := I - 1;
        TP := I;
      END
    ELSE
      TP := 0;
      IF I < 249
      BEGIN
        WHILE(DTAB[I].SPL = DTAB[I+1].SPL) AND (I < BP) DO
          I := I + 1;
          BP := I;
        END
      ELSE
        BP := 249;
      END;
    END;
  END;

```

```

PROCEDURE FOND;
BEGIN
  BLKPAGE;
  WRITELN(' RF PRF SCAN TYPE SCAN');
  WRITELN(' RF, PRF, SCAN, SCANTYPE');
  WRITELN;
  WRITELN('NOT FOUND');
  WRITELN;
  WRITELN;
END;

```

```

PROCEDURE FCONT;
VAR J,K:INTEGER;
BEGIN
  RESET('FILE');
  J := FOUND * 2 + 1;
  K := 2;
  WHILE K < J DO
    BEGIN
      READLN('FILE');
      K := K + 1;
    END;
  BLKPAGE;
  WRITELN('ELINT MC NATO NICKNAME FC CDS COMMENTS');
  J := 2;

```

```

WHILE J < 7 DO
BEGIN
  READ(DFILE,C);
  WRITE(C);
  J := J + 1;
END;
READ(DFILE,C);
WRITE(C);
READ(DFILE,C);
WRITE(C,' ');
J := 0;
WHILE J < 16 DO
BEGIN
  READ(DFILE,C);
  WRITE(C);
  J := J + 1;
END;
READ(DFILE,C);
WRITE(C);
READ(DFILE,C);
WRITE(C,' ');
J := 0;
WHILE J < 5 DO
BEGIN
  READ(DFILE,C);
  WRITE(C);
  J := J + 1;
END;
J := 0;
WHILE J < 26 DO
BEGIN
  READ(DFILE,C);
  WRITE(C);
  J := J + 1;
END;
READLN(DFILE);
WRITELN;
WRITELN;
WRITELN('ARE YOU READY TO CONTINUE? (Y/N)');
READ(C);
WHILE C <> 'Y' DO
  READ(C);
END;

```

```

PROCEDURE LTAB;
VAR I,SEGL,SEGH: INTEGER;
    DISKL,DISKH: INTEGER;
BEGIN
  RESET(TABLE,'REFTAB.TEXT');
  READLN(TABLE,DISKL,DISKH);
  IF (RF >= DISKL) AND (RF <= DISKH) THEN
  BEGIN
    READLN(TABLE,SEGL,SEGH,SIZE);
    I := 0;
  END;

```

```

WHILE SIZE <> 999 DO
BEGIN
  I := I + 1;
  IF(RF >= SEGL) AND (RF <= SEGH) THEN
  BEGIN
    BP := SIZE - 1;
    SIZE := 999;
  END
  ELSE
  BEGIN
    READLN(TABLE,SEGL,SEGH,SIZE);
  END;
END;
CASE I OF
  1: SEGMFY := 'SEG1.TEXT';
  2: SEGMFY := 'SEG2.TEXT';
  3: SEGMFY := 'SEG3.TEXT';
  4: SEGMFY := 'SEG4.TEXT';
  5: SEGMFY := 'SEG5.TEXT';
  6: SEGMFY := 'SEG6.TEXT';
  7: SEGMFY := 'SEG7.TEXT';
  8: SEGMFY := 'SEG8.TEXT';
END;
RESET(DFILE,SEGMFY);
END
ELSE
BEGIN
  BLKPAGE;
  NFOCN;
  WRITELN('DO YOU HAVE ANOTHER DISK? (Y/N,');
  READ(C);
  CHECK;
  IF C = 'Y' THEN
  BEGIN
    BLKPAGE;
    WRITELN('LOAD THE DISK INTO THE LEFT SIDE DISK');
    WRITELN('DRIVE WITH LABEL FACING YOU AND DOWN. ');
    WRITELN;
    WRITELN;
    WRITELN('HAVE YOU DONE THIS? (Y/N)');
    READ(C);
    WHILE C <> 'Y' DO
      READ(C);
    CLOSE(TABLE,LOCK);
    LTAB;
  END
  ELSE
    EXIT(SEARCH);
  END;
END;

PROCEDURE LDTAB;
VAR I: INTEGER;
BEGIN

```

```

I := 0;
WHILE I <= BP DO
BEGIN
  READ(DFILE,DTAB[I].RFL,DTAB[I].RFH,DTAB[I].PRFL);
  READ(DFILE,DTAB[I].PRFH,DTAB[I].SPL,DTAB[I].SPH);
  READ(C);
  READLN(DTAB[I].ST);
  READLN(DFILE);
  I := I + 1;
END;
END;

PROCEDURE ENTRY(I:INTEGER);
BEGIN
  CASE I OF
    1: BEGIN
      BLKPAGE;
      WRITELN('ENTER PRF(XXXXX) <rtm>');
      READLN(PRF);
      END;
    2: BEGIN
      BLKPAGE;
      WRITELN('ENTER RF(XXXXX) <rtm>');
      READLN(RF);
      END;
    3: BEGIN
      BLKPAGE;
      WRITELN('ENTER SCAN(XXXX) <rtm>');
      READLN(SCAN);
      END;
    4: BEGIN
      BLKPAGE;
      WRITELN('ENTER TYPE SCAN(AAAAA) <rtm>');
      READLN(SCANTYPE);
      END;
  END;
END;
END;

PROCEDURE HUNT;
BEGIN
  WHILE (TP <= BP) AND (KEEP = TRUE) DO
  BEGIN
    POINT := (BP-TP +1) DIV 2 + TP;
    IF(DTAB[POINT].RFL <= PRF) AND (DTAB[POINT].RFH >= RF) THEN
    BEGIN
      GROUPPRF(POINT);
      POINT := (BP-TP +1) DIV 2 + TP;
      IF(DTAB[POINT].PRFL <= PRF) AND (DTAB[POINT].PRFH >= PRF)
      THEN
      BEGIN
        GROUPPRF(POINT);
        POINT := (BP - TP + 1) DIV 2 + TP;
        IF(DTAB[POINT].SPL <= SCAN) AND (DTAB[POINT].SPH >= SCAN)
        THEN

```

```

BEGIN
  GROUPSCAN(POINT);
  IF TP <> BP THEN
    BEGIN
      WHILE TP <> BP DO
        BEGIN
          IF (DTAB[TP].ST = SCANTYPE) THEN
            BEGIN
              FOUND := TP;
              KEEP := FALSE;
            END
          ELSE
            TP := TP + 1;
          END;
        END
      ELSE
        BEGIN
          IF DTAE[TP].ST = SCANTYPE THEN
            BEGIN
              FOUND := TP;
              KEEP := FALSE;
            END
          ELSE
            TP := TP + 1;
          END;
        END
      END
    ELSE
      BEGIN
        IF DTAE[POINT].SPL <= SCAN THEN
          TP := POINT
        ELSE
          BP := POINT - 1;
        END;
      END
    ELSE
      BEGIN
        IF DTAE[POINT].PRFL <= PRF THEN
          TP := POINT
        ELSE
          BP := POINT - 1;
        END;
      END
    ELSE
      BEGIN
        IF DTAE[POINT].PFL <= RF THEN
          TP := POINT;
        ELSE
          BP := POINT - 1;
        END;
      END;
    END;
  END;
  BEGIN (* SEARCH *)
    FOUND := 0;

```

```

BLKPAGE;
WRITELN('ARE YOU READY TO SEARCH? (Y/N)');
READ(C);
WHILE C <> 'Y' DO
  READ(C);
  ENTRY(1);
  ENTRY(2);
  ENTRY(3);
  ENTRY(4);
  KEEP := FALSE;
  WHILE KEEP = FALSE DO
    BEGIN
      BLKPAGE;
      WRITELN('YOUR ENTRIES ARE:');
      WRITELN;
      WRITELN(' PRF  RF  SCAN  TYPE SCAN');
      WRITELN(PRF, ' ', RF, ' ', SCAN, ' ', SCANTYPE);
      WRITELN;
      WRITELN('DO YOU WISH TO CHANGE ANY? (Y/N)');
      READ(C);
      CHECK;
      IF C = 'Y' THEN
        BEGIN
          WRITELN;
          WRITELN('WHICH ONE DO YOU WANT TO CHANGE');
          WRITELN('(PRF,RF,SCAN,TYPE SCAN) <rtn>');
          READLN(C);
          IF C = 'P' THEN ENTRY(1)
          ELSE
            IF C = 'R' THEN ENTRY(2)
            ELSE
              IF C = 'S' THEN ENTRY(3)
              ELSE
                IF C = 'T' THEN ENTRY(4)
                ELSE
                  ERROR;
        END
      ELSE
        KEEP := TRUE;
    END;
  LTAB;
  LD TAB;
  HUNT;
  IF KEEP = TRUE THEN
    BEGIN
      FOUND;
      WRITELN('ARE YOU READY TO CONTINUE? (Y/N)');
      READ(C);
      WHILE C <> 'Y' DO
        READ(C);
      END
    ELSE
      FOUND;
    END;
  END;

```

D. PROCEDURE TALK

PROCEDURE TALK;

PROCEDURE CONTINUE;

BEGIN

GOTOXY(0,20);

WRITELN('ARE YOU READY TO CONTINUE? (Y/N)');

READ(C);

WHILE C <> 'Y' DO

READ(C);

END;

PROCEDURE PAGE1;

BEGIN

BLKPAGE;

WRITELN('THE EMITTER IDENTIFIER WORKS BY USING THE ');

WRITELN('PARAMETERS TAKEN FROM THE CURRENT DETECTION ');

WRITELN('UNIT. THE PARAMETERS ARE THEN ENTERED INTO ');

WRITELN('IDENTIFIER BY THE OPERATOR THROUGH THE KEYBOARD.');

WRITELN;

WRITELN('THE KEYBOARD ENTRIES ARE REQUESTED WHEN THE ');

WRITELN('OPERATOR HAS RESPONDED TO THE QUESTION, 'ARE YOU');

WRITELN('READY TO SEARCH? (Y/N)'. THE TWO TYPES OF ENTRIES');

WRITELN('ARE ONE, A SIMPLE YES OR NO. TO DO THIS RESPONSE, ');

WRITELN('MERELY DEPRESS THE KEY LABELED 'Y' OR 'N' WHEN ');

WRITELN('PRESENTED WITH A QUESTION ENDING IN (Y/N).');

CONTINUE;

END;

PROCEDURE PAGE2;

BEGIN

BLKPAGE;

WRITELN('THE OTHER TYPE OF RESPONSE IS FOLLOWED BY DE-');

WRITELN('PRESSING THE KEY LABELED 'RETURN'. LIKE THESE:');

WRITELN;

WRITELN('ENTER PRF(XXXXX) <rtm> ');

WRITELN('ENTER PF(XXXXX) <rtm> ');

WRITELN('ENTER SCAN(YXXXX) <rtm> ');

WRITELN('ENTER SCAN TYPE(AAAAA) <rtm> ');

WRITELN;

WRITELN('FIRST, THE X'S REPRESENT NUMBERS AND THE A'S RE-');

WRITELN('PRESENT CHARACTERS. THE NUMBER OF X'S OR A'S');

WRITELN('REPRESENT THE MAXIMUM NUMBER OF POSITIONS ALLOWED');

WRITELN('IN THE RESPONSE. SIMPLY ENTER EXACTLY WHAT IS ');

WRITELN('DISPLAYED ON THE CURRENT DETECTION UNIT, FOLLOWED');

WRITELN('BY THE RETURN <rtm> KEY. ');

CONTINUE;

END;

PROCEDURE PAGE3;

BEGIN

BLKPAGE;

WRITELN('IF YOU HAVE MADE A MISTAKE IN THE ENTRY, BEFORE');

WRITELN('DEPRESSING THE RETURN KEY, ALL YOU HAVE TO DO IS ');

```

WRITELN('TO DEPRESS THE KEY LABELED WITH A LEFT POINTING ');
WRITELN('ARROW ON IT. THEN MAKE THE CORRECT RESPONSE. ');
WRITELN('IF, HOWEVER, YOU DID PRESS THE RETURN KEY, YOU ');
WRITELN('WILL BE ABLE TO CORRECT IT AT THE VERIFICATION. ');
CONTINUE;
END;

```

PROCEDURE PAGE4;

```

BEGIN
  BLKPAGE;
  WRITELN('TO VERIFY YOUR ENTRIES THE FOLLOWING MESSAGE APPEARS: ');
  WRITELN;
  WRITELN('      YOUR ENTRIES ARE: ');
  WRITELN;
  WRITELN('      PFF      RF      SCAN      SCAN TYPE ');
  WRITELN('      XXXXX    XXXXX    XXXXX    AAAAA ');
  WRITELN;
  WRITELN('DO YOU WANT TO CHANGE ANY? (Y/N) ');
  WRITELN;
  WRITELN('IF YOU RESPOND YES. THE FOLLOWING MESSAGE APPEARS: ');
  WRITELN;
  WRITELN('WHICH ONE DO YOU WANT TO CHANGE ');
  WRITELN('PFF,RF,SCAN,TYPE SCAN)? <ptr> ');
  WRITELN;
  WRITELN('YOU SIMPLY ENTER EITHER PFF, OR RF, OR SCAN, ');
  WRITELN('OR TYPE SCAN FOLLOWED BY THE RETURN KEY. THE ');
  WRITELN('IDENTIFIER WILL AGAIN ASK FOR THAT ENTRY. ');
  CONTINUE;
END;

```

PROCEDURE PAGE5;

```

BEGIN
  BLKPAGE;
  WRITELN('THE OUTPUT OF THE IDENTIFIER IS ONE OF THE ');
  WRITELN('TWO FOLLOWING FORMS: ');
  WRITELN;
  WRITELN('ELINT MC NATO NICKNAME FC CDS NR. COMMENTS ');
  WRITELN('AAAAA X AAAAAAAAAAAAAA AA XXXX AAAAAAAAA ');
  WRITELN('AAAAA X AAAAAAAAAAAAAA AA XXXX AAAAAAAAA ');
  WRITELN;
  WRITELN('OR ');
  WRITELN;
  WRITELN('PFF      RF      SCAN      SCAN TYPE ');
  WRITELN('XXXXX    XXXXX    XXXXX    AAAAA ');
  WRITELN('NOT FOUND ');
  CONTINUE;
END;

```

BEGIN (* START OF PROCEDURE TALK *)
PAGE1;
PAGE2;
PAGE3;
PAGE4;
PAGE5;
END; (* END OF PROCEDURE TALK *)

E. MAIN PROGRAM

```
PROGRAM ID;  
VAR C: CHAR;  
    LOOP: BOOLEAN;
```

```
PROCEDURE EFPFOP;  
BEGIN  
    WRITELN;  
    WRITELN('INCORRECT RESPONSE');  
    WRITELN('PLEASE, TRY AGAIN');  
    READ(C);  
END;
```

```
PROCEDURE CHECK;  
BEGIN  
    WHILE C<>'Y' AND C<>'N' DO  
        ERROR;  
END;
```

```
PROCEDURE BLKPAGE;  
VAR CNT: INTEGER;  
BEGIN  
    CNT:= 0;  
    WHILE CNT < 26 DO  
        BEGIN  
            WRITELN;  
            CNT:= CNT + 1;  
        END;  
    GOTOXY(2,0);  
END;
```

```
PROCEDURE TALK;  
(SEE APPENDIX B ITEM 2)
```

```
PROCEDURE FPMAT;  
(SEE APPENDIX B ITEM 1)
```

```
PROCEDURE SORT;  
(SEE APPENDIX B ITEM 3)
```

```
PROCEDURE SEARCH;  
(SEE APPENDIX B ITEM C)
```

```
BEGIN (* MAIN PROGRAM *)  
    LOOP := TRUE;  
    BLKPAGE;  
    WRITELN('WELCOME TO THE ELECTRONIC EMITTER IDENTIFIER. ');  
    WRITELN;  
    WRITELN('HAVE YOU USED ME BEFORE? (Y/N) ');  
    READ(C);  
    CHECK;  
    IF C = 'N' THEN TALK  
        ELSE
```

```

BEGIN
  BLKPAGE;
  WRITELN('DO YOU NEED A REFRESHER? (Y/N) ');
  READ(C);
  CHECK;
  IF C = 'Y' THEN TALK;
  END;
  BLKPAGE;
  WRITELN('DO YOU HAVE A DISK LABELED DATA? (Y/N) ');
  READ(C);
  CHECK;
  IF C = 'Y' THEN SEARCH
  ELSE
    BEGIN
      REWAT;
      SOFT;
    END;
  WHILE LOOP = TRUE DO
    SEARCH;
  END. (* MAIN PROGRAM *)

```

BIBLIOGRAPHY

1. Naval Electronic Engineering Office Damneck, Command Presentation, 1979.
2. Bush, R.F.J., "Soviet ECM Capabilities Detailed", Electronic Warfare and Defense Electronics, Dec. 1978, 60-68.
3. Fawcett, Dr. C.W., "Passive Warfare for Destroyers", Navsea Journal, Mar. 1977, 24-29.
4. Hartman, R., "EW Essentials for FPB's", Electronic Warfare and Defense Electronics, Jan. 1979, 55-67.
5. Patch, B., "Computer Use in Electronic Warfare", Microwave Journal, Sept. 1979, 53-67.
6. Hellerman, H., Digital Computer System Principles, McGraw-Hill, 1973.
7. Horowitz, E. and Sahni, S., Fundamentals of Computer Algorithms, Computer Science Press, Inc., 1978.
8. Institute for Information Systems, UCSD (Mini-Micro Computer) PASCAL Reference Manual, 1978.
9. Smith, S.W., Operational EA-6B Mission Planning Programs, Master's Thesis, Naval Postgraduate School, 1978.
10. Godley, J.P., Microcomputer Based Generator of Recurring Operational Reports, Master's Thesis, Naval Postgraduate School, 1977.

DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 2142 Naval Postgraduate School Monterey, California 93940	2
3. Professor Grodon H. Bradley, Code 525z Department of Computer Technology Naval Postgraduate School Monterey, California 93940	1
4. LTCR Frank Burkhead, Code 523g Department of Computer Technology Naval Postgraduate School Monterey, California 93940	1
5. Lt. Gary L. Bush 924 North Tison Street Gainesville, Texas 76242	1
6. Professor W.F. Schneidewind, Code 528s Department of Computer Technology Naval Postgraduate School Monterey, California 93940	1