(12)

AD A114524

DYNAMIC TUNING OF A SIGNAL SORTER IN A DENSE ENVIRONMENT

A.M. ABDALLA
THE GEORGE WASHINGTON UNIVERSITY

February 10, 1982

A final report for
NRL Grant N00014-80-C-0572

DTIC FILE COPY

82 05 14 050

CONTENTS

A. Introduction

B. Background Information

1. Environment Simulation

2. System Configuration
   a) Receiver and Antenna Model
   b) Processing Hardware
      1) LIST Forming Processor
      2) LOAD CAM Processor
      3) Microprocessor Array

C. Architecture Tuning

1. LIST Reconfiguration
   a) FIFO Implementation
   b) RAM Implementation

D. Environmental Inputs and Parameters of the System Components

1. Environmental Affects
   a) Separation Time
   b) Frequency Hoppers

2. List Configuration

3. Associative Processor

4. Advance Load Time

5. Variable Environments

6. Long-Run Analysis
   a) Environment Model: Special Problems
   b) LIST Sizes
   c) Buffer Sizes
   d) Rate of Nomatches

E. Conclusions

F. Appendix

1. Emitter Parameters - Maximum and Minimum Values
2. Statistical Package
3. Emitter Environment - 300 Emitters
4. Emitter Environment - 100 Emitters
5. Results 300-Emitter Stationary Environment
6. Frequency Hopper Run Results
7. Program Listing and Flowcharts

A. INTRODUCTION

This report is in reference to the research on "Dynamic Tuning of a Signal Sorter in a Dense Environment" sponsored by the Naval Research Laboratory. 7/1/80. principal investigator: Dr. A.M. Abd-Alla of the George Washington University.

The purpose of the research was to study the application of dynamic tuning, if practically feasible, to the design of a particular signal sorter in order to improve its performance. A simulation model of a signal sorter in a dense environment, which was studied and developed through work by the Advanced Techniques Branch of the TEW Division of the NRL was used for problem analysis.

Signal sorting involves the correlation of the mass of signals detected by a receiver with the individual sources that generate each signal. The signal sorter receives a pulse train which contains the signals of many different sources and which must be measured and analyzed to recognize the sources of the signals, the emitters, and to separate the interleaved pulses into individual emitters.

Identification of particular emitters necessitates maintenance of a file to compare received pulses with those currently identified or known to be in a particular environment and to update when new emitters are detected in the environment. The signal sorter generates these files based on (1) parameters measured by the receiver, such as direction-of-arrival [DOA], carrier-frequency [CF], pulse-width [PW] and latest time-of-arrival [TOA], and (2) generated parameters based on measured parameters, such as pulse-repetition-interval [PRI].

In addition, the signal sorter must be able to detect some irregularities and/or intentional variations in the signals received, such as signal dropout, signal overlap or measurement inconsistencies. A dense environment requires a signal sorter with high throughput rates in order to keep up with the high data rate for real-time requirements; and for this particular application (airborne), size, weight and power consumption must also be kept to a minimum.

B. BACKGROUND INFORMATION

The simulation model includes models of the environment, the antenna and receiver, and the signal identification and sorting system. A summary description of that model is presented here.[1]

B.1. Environment Simulation

Two subroutines, NDAGE and NEMIT, simulate the electromagnetic activity seen by the receiver. NDAGE generates the initial parameters for each emitter: see figure 1 for field definitions for each parameter of the emitter array. It randomly selects a value for each parameter out of a range of expected values for that particular parameter. NEMIT modifies and updates

FIELD DEFINITION OF THE PARAMETER ARRAY FOR EACH EMITTER:
EMITR(I,J) WHERE I=EMITTER NUMBER, J=DEFINED BELOW

1. DX - X DISPLACEMENT OF EMITTER AND SIGNAL SORTER (METERS)
2. DY - Y DISPLACEMENT OF EMITTER AND SIGNAL SORTER (METERS)
3. POWER OF EMITTER (INCLUDES ANTENNA GAIN)
4. MAINLOBE SIZE - OF EMITTER IN DEGREES
5. SIDELOBE LOSS - IN DECIBELS DOWN FROM MAIN BEAM GAIN
6. MAX. ANTENNA ANGLE - UP SCAN LIMIT OF ANTENNA IN DEGREES
7. SCAN RATE - IN HZ
8. PRI - PULSE REPETITION INTERVAL IN SECONDS
9. PULSE WIDTH - IN SECONDS
10. FREQUENCY - IN GIGAHZ
11. ON TIME - TIME WHICH THE EMITTER IS TURNED ON (BEFORE MAXIMUM ON-TIME [EMS])
12. OFF TIME - TIME AT WHICH THE EMITTER IS TURNED OFF
13. RCVR POWER - POWER LEVEL FROM EMITTER I SEEN AT THE RECEIVER ANTENNA
14. TOA - TIME OF ARRIVAL OF TRANSMITTED PULSE AT RCVR ANTENNA
15. DOA - DIRECTION OF ARRIVAL OF TRANSMITTED PULSE AT RCVR ANT.
16. FLAG - SET FOR DURATION OF PULSE
17. INITIAL ANTENNA ANGLE - IN DEGREES
18. DZ - Z DISPLACEMENT OF EMITTER AND SIGNAL SORTER (METERS)
19. MIN. ANTENNA ANGLE - LOWER SCAN LIMIT IN DEGREES
20. VX - X VELOCITY COMPONENT OF SIGNAL SORTER PLATFORM
21. VY - Y VELOCITY COMPONENT OF SIGNAL SORTER PLATFORM
22. VZ - Z VELOCITY COMPONENT OF SIGNAL SORTER PLATFORM
23. TYPE: -1:MOVING EMITTER, 0:FIXED, +1:COLLISION COURSE
24. SPARE
25. SPARE

Figure 1. Field Definition for Each Emitter

the time-varying parameters for each emitter on a pulse-to-pulse basis.

The environment simulation generates interleaved pulse trains representing several types of emitters that may be encountered in a real environment. Regular emitters, those with no intentional variation of their parameters, predominate, although exotic emitters may also be generated. A variance on some parameters can also be added on a pulse-to-pulse basis to account for measurement irregularities, parameter agility, or emitter drift.

Flexibility is included in the model to permit varying environment parameters, such as emitter PRI distribution, emitter location, signal densities, etc., in different runs and to allow the same environment to be tested against several signal sorter designs. See appendix F.1 for the maximum and minimum values for each parameter.

B.2. System Configuration

B.2.a. Antenna & Receiver

The subroutine RCVR simulates the antenna and receiver model. The antenna model is a phased array antenna with a beam forming network . This antenna is equivalent to several fixed directional antennas all integrated into an array covering the full 360° view around the signal sorter platform.

The receiver model is comprised of 16 video detectors, one at each beam port of the antenna. Amplitude measurements from these detectors are used to calculate direction of arrival. The receiver also measure PW, CF and TOA. These values are then digitized and passed to the processing system for identification and sorting.

The DOA is used as a sorting parameter. Each beam is divided into four (4) distinct bins, creating 64 bins for the 360° view. Each emitter is filed and accessed in main memory by its bin number. A match or no-match with the file in main memory is based on finding the matching frequency parameter for a particular emitter in its DOA bin. TOA is used to calculate pulse-repetition-interval (PRI) for prediction of subsequent pulse arrivals. Values for antenna or receiver sensitivity, bandwidth, gains, etc., were chosen within the range of current technologies.

B.2.b. Processing Hardware

Configuration of the signal sorting system includes both pipeline and parallel architectures. A block diagram of the system is shown in figure 2. The CAM, the Processor-Array buffer, the LIST buffer and each bin (128 of them) of the CAM-LOAD-LIST, are First-In-First-Out (FIFO) buffers to synchronize the rate of the data flow.

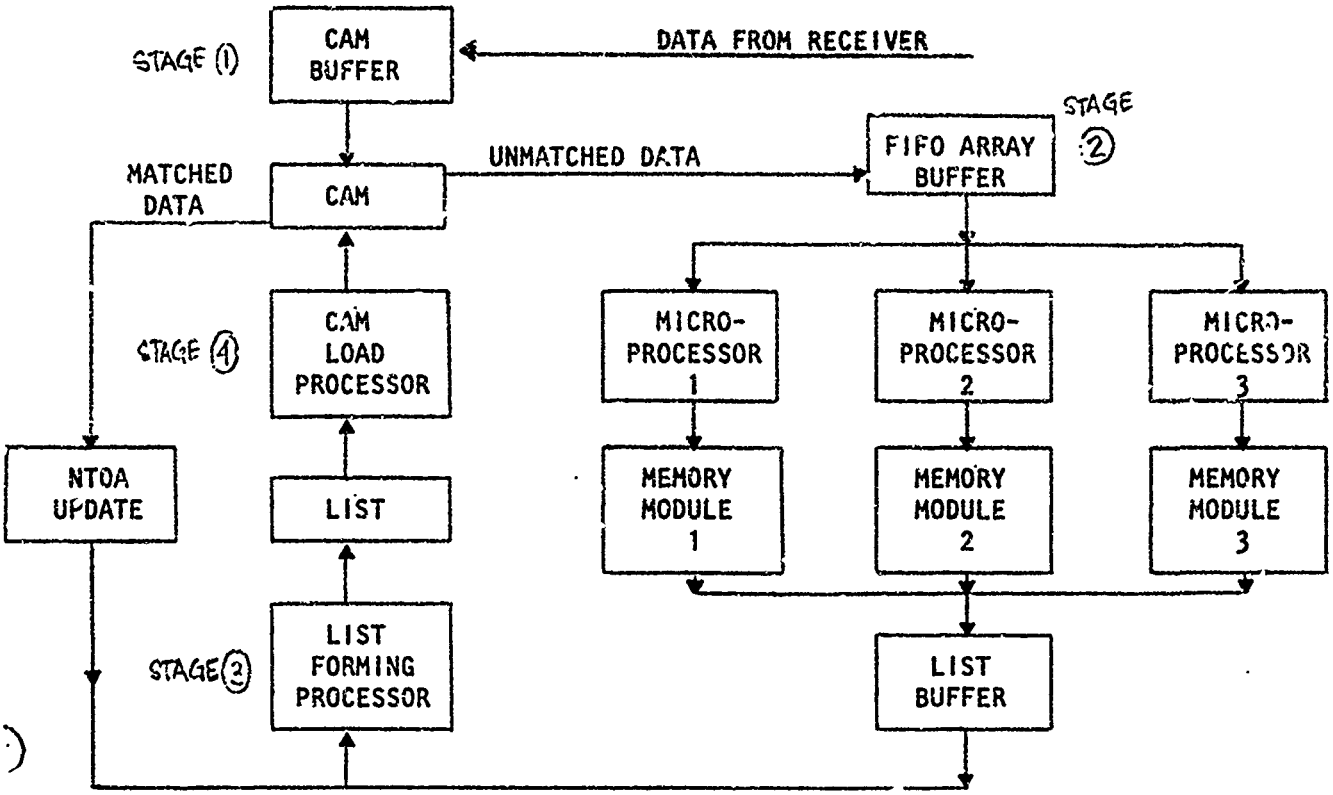


Figure 2. Signal Sorting System

The signal sorter task consists of the following stages: (1) presort, (2) identification and file generation, (3) list forming based on the expected arrivals and (4) CAM load.

Presort utilizes the CAM, which is continuously updated based on active emitters detected by the receiver. A CAM word contains the DOA and CF of an active emitter. The PRI of the emitter is stored in the corresponding word of the PRI memory, as shown in figure 3. The received data are compared to the CAM for a match. If no match occurs, the data is passed down the pipeline to the 2nd stage. This CAM organization uses the CAM to provide a fast response to the incoming signals and, hence, can handle them on a pulse-to-pulse basis.

Data that is matched by the CAM is updated for next-time-of-arrival [NTOA] and passed to stage 3, LIST-forming of expected arrivals. This LIST is ordered based on NTOA, and emitter parameters in the LIST are loaded into the CAM when the real-time clock matches the NTOA of the emitter. So the CAM acts as a filter of the data stream to stage 2, (identification and file generation), and thereby reduces the the data stream to that stage by passing only unmatched data. This presort is done in the simulation subroutine ASSOC.

The unmatched data is compared to the file of previously identified emitters in stage 2 and results in either an addition of a new emitter to the file or updating the parameters of an existing emitter if the unmatched data matches that of an existing file within predetermined limits. These emitters, once identified by the array processors, are stored in a LIST buffer, which will be used to load the CAM LOAD LIST. These tasks are carried out in the MPPR subroutine.

The CAM LOAD LIST of stage 4 is used to load the CAM, and the CAM is loaded only with the nearest expected arrivals, which allows the CAM to remain small. This is necessary since the current size of CAMs (content addressable memories) is limited and is restricted to 24 registers in this simulation. In a dense environment, the CAM would not be able to hold data on all the emitters at once. Data for an emitters is loaded into the CAM based on the expected arrival time of the next pulse. Thus, the CAM is continuously loaded. The algorithm for determining the loading time for emitter data into the CAM is discussed in the next section.

B.2.b.1. LIST Forming Processor

The CAM LOAD LIST is a series of FIFOs which are ordered as a series of time slots, the time slots correlated with the Next-Time-of-Arrival (NTOA) of an emitter. Data for an emitter is loaded into the FIFO corresponding to the NTOA of that emitter. The correlation of NTOA is determined by the value of some middle bits of the NTOA which represents the "time slot" bits, the position of these bits being a function of the distribution of the PRIs of the emitters. This provides a uniform distribution of emitters throughout the FIFOs.

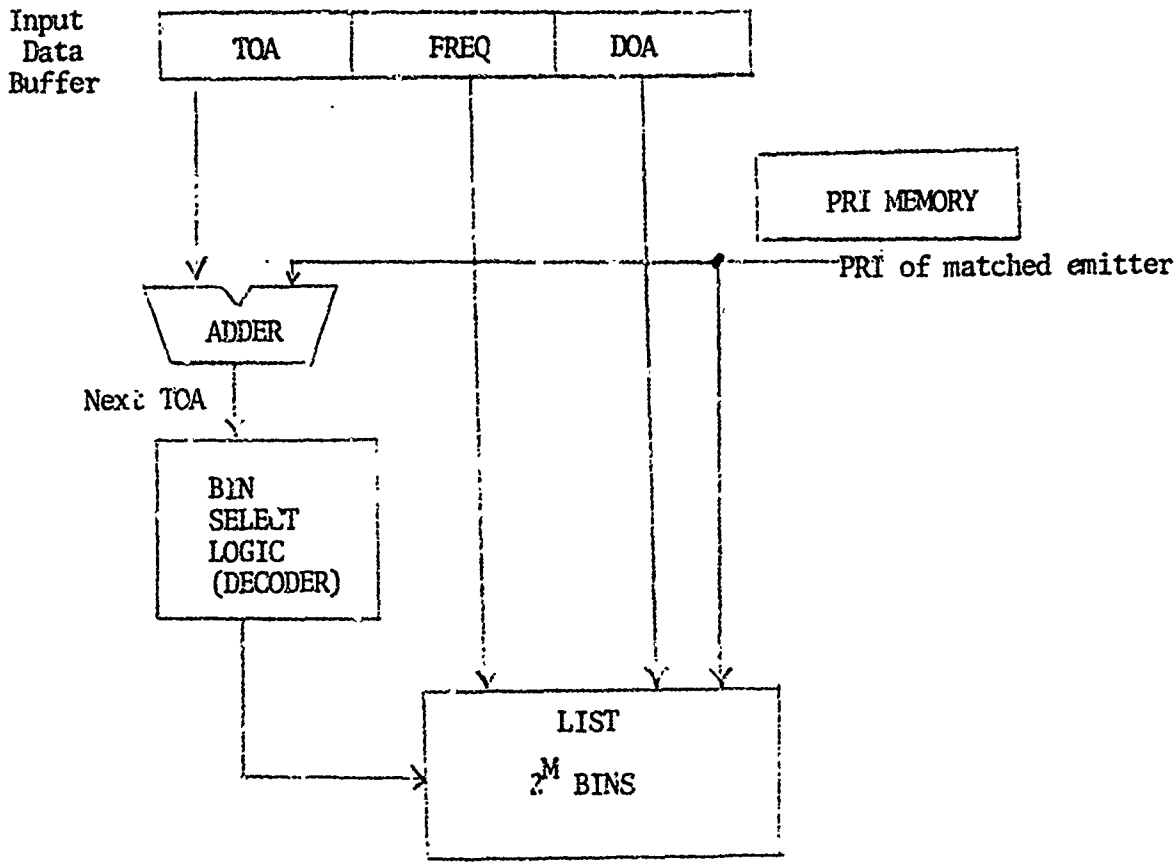


Figure 3. Data Pass for Matched Emitters

Given a uniform distribution of PRIs over a known range, based on the highest and lowest PRI of the emitter environment, the time window for each FIFO is based on the following:

NTOA WORD

M = # of middle bits used as a time slot (M>0)

N = # of bits to the right of the time slot in the NTOA word (N>0). Each word must be shifted N bits to the right to position the M "time slot" bits as the least significant bits of the word.

In the simulation runs, the value of the least significant bit of the NTOA is one microsecond. The M value determines the number of FIFOs used for the LIST, i.e. 2^M FIFOs. In the current simulation, M is constrained to be less than or equal to 8; therefore, the maximum number of FIFOs comprising the LIST is 2^8 or 256. Determination of the values of M and N based on the environment is discussed later.

Data may be loaded into the LIST from the associative processor (the CAM stage) or from the microprocessor array. Data from the associative processor are emitters which matched in the CAM and will be arriving again in a time period equal to the particular emitter's PRI. The NTOA of the matched emitter is computed by adding the PRI to the TOA, and the emitter data is loaded into the LIST again in the FIFO corresponding to the new NTOA time window, as shown in figure 3. Data from the microprocessor array are those emitters which were unmatched in the CAM and have now been matched and updated or added to the file as new emitters. This data is added into a buffer for the LIST and loaded into the LIST at a convenient time.

So, the LIST Forming Processor creates an ordered list of those emitters whose pulses will arrive next. This function is simulated in the LCAM subroutine.

B.2.b.2. LOAD CAM Processor

The LOAD CAM Processor loads the CAM from the LIST FIFOs shortly before the next time of arrival of those emitters. However, the CAM is loaded only when it is not doing a search. Processing data from the receiver has higher priority than loading. A real-time clock determines the particular FIFO in the LIST from which data will be loaded into the CAM. The FIFO is correlated to the real-time slot by the M and N values discussed earlier.

Consequently, the CAM is loaded only with those emitters that will arrive during the next time period. The emitter data is loaded into the CAM during the same time slot as the NTOA of the emitter. So the particular FIFO in the LIST to be unloaded

is determined the same way as the FIFO to be loaded is selected. Data within a FIFO (bin) is unloaded sequentially on a first-in-first-out basis. The LOAD CAM Processor function is simulated in the CAM and LCAM subroutines.

B.2.b.3. Microprocessor Array (MA)

Data that is not matched in the CAM is passed to a FIFO buffer for the microprocessor-array which compares the unmatched data to its files for a match on frequency in three adjacent DOA bins. New data is stored in the microprocessor-array (MA) memory based on its DOA. There are 64 modules in the MA memory to correspond to the 64 DOA cells of the receiver system. Each module contains a number of entries (e.g., 16). Each entry includes the DOA, the frequency, the pulse-width, the TOA, the PRI, the type of emitter, and a status flag for number of pulses received.

A NOMATCH in the CAM passes that particular data word to the MA input buffer. Because some of these nomatches are caused by a drift in an emitter's DOA, a search is done on three modules in the MA memory to attempt to find the emitter in an adjacent module. The microprocessor-array consists of three microprocessors operating in parallel. A search for a frequency match is done on the DOA module corresponding to that of the received emitter and also on the modules with DOA on either side of it (i.e., DOA+1 and DOA-1). When a match is found in an adjacent module, the emitter data is transfered into the proper module (the most recent value of DOA) and the old entry is purged.

The three processors search the three memory modules in parallel. When one processor finds a match on the emitter, it interrupts the other two processors. The three processors then continue processing the next entry from the input buffer until the buffer is empty. In addition to searching for a DOA drift, the MA also searches for a frequency drift in each module within +1 Hz. So the MA does a between limits match on both the frequency and the DOA. Data that is matched in this search is updated in memory and passed to an output buffer for later loading into the LIST. The MA function is simulated in the MPPR subroutine.

C. ARCHITECTURE TUNING

Tuning as it applies to a fixed system architecture refers to a computer system structure that has been adjusted to solve a particular problem more efficiently. In the case of a signal sorter system, a static architecture limits its effectiveness to a range of emitter parameters generated by the envirnment model because the system is tuned to operate most efficiently in that environemnt. These parameters are uniformly distributed and random; they are also bounded by arbitrary upper and lower limits based on previously observed values. In order to make the system more flexible, i.e., to expand the variety of environments in which the signal sorter can operate efficiently, requires dynamic

adjustments in the architecture.

Dynamic tuning is adjustment that occurs during operation of the system whenever changes in environment data require it. Since parameters such as environment density, total number of pulses per second, distribution of PRIs and DOA distributions affect system performance, changes in the processing system to accommodate these changes in the environment could provide acceptable system performance over a wider range of input characteristics. In the next section, the effect of the PRI's range on the previously described LIST structure is analyzed.

C.1. LIST Reconfiguration

It has been shown[2] that improved performance results when the number of modules in the LOAD CAM LIST is chosen based on the PRI distribution. This assumes a uniform distribution of PRI values over the range of PRIs. Reconfiguration of the LIST is based on the PRI distribution of the environment and the loading time slot parameters M and N such that:

(1) $2^N <$ Minimum PRI
(2) $2^M \geq$ Maximum PRI

The first condition provides a good distribution of emitters over the number of bins and the second condition covers the entire range of NTOA, including emitters with maxium PRI. This provides that the time to cycle through all the bins in the LIST is greater than the maximum PRI.

Since M determines the number of modules and keeping M small is a consideration, the upper limit on M must be restricted. For LIST configuration, the following conditions should be true:

1. $M \leq$ M-max (log2 of max # of modules allowed)
2. $N < [\log_2 PRIL]$ []=integer value
3. $N+M=[\log_2 PRIH]+1$ of quantity

For a summary of configurations of (M,N) giving the best results for values of (PRIH,PRIL) see table 1 and figure 4. The maximum value for 2^M is chosen arbitrarily to be 128.

Reconfiguration of the LIST is implemented by monitoring the maximum and the minimum PRIs in the environment that is being tested. The microprocessor array is the monitor which outputs new values to the LIST control hardware to initiate reconfiguration. This is done in subroutine CONFIG, which is called by MPPR.

In simulation runs in this report, LIST configuration was static if "number of emitters identified before configuration", an input parameter, was greater than the number of emitters in the environment, and dynamic if that input parameter was less than the number of emitters present.

The range of PRIs affects the structure of the LIST used to load the CAM with the data for next expected pulse arrivals.

TABLE 1. Configuration vs PRI Distribution

MAX PRI (PRI_H) / MIN PRI (PRI_L)	2000	2048	2500	3072	4000	4096	5118	7168	8000
128	5,6	6,6	6,6	6,6	6,6	6,7	6,7	6,7	6,7
256	5,6	6,6	6,6	5,7	5,7	6,7	6,7	5,8	5,8
384	5,6	6,6	6,6	5,7	5.7	6,7	6,7	5,8	5,8
512	5,6	6,6	6,6	5,7	5,7	6,7	6,7	5,8	5,8
640	5,6	5,7	5,7	5,7	5,7	6,7	6,7	5,8	5,8
768	5,6	5,7	5,7	5,7	5,7	6,7	5,8	5,8	5,8
896	5,7	5,7	5,7	5,7	5,7	6,7	5,8	4,9	4,9
1024	5,7	5,7	5,7	5,7	5,7	5,8	5,8	4,9	4,9

(M,N = Log_2 (number of Bins),relative Bin size)

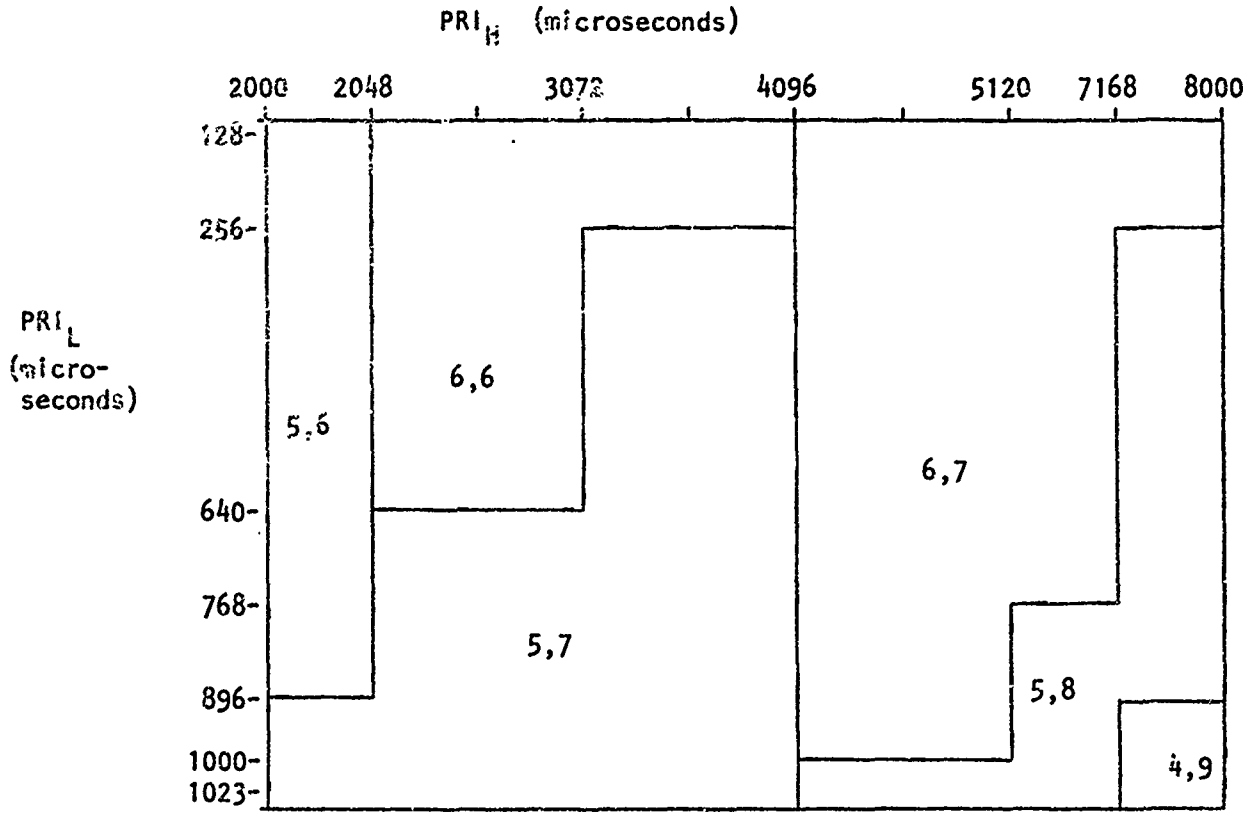


Figure 4. (M,N) LIST Configuration vs PRI Distribution

thus the LIST organization (the size of each module and the number of modules), can be based on the minimum and maximum PRI, with the total size of the LIST remaining fixed. Implementations of modules of the LIST structure using both FIFO and RAM memories are discussed below.

C.1.a. FIFO Implementation

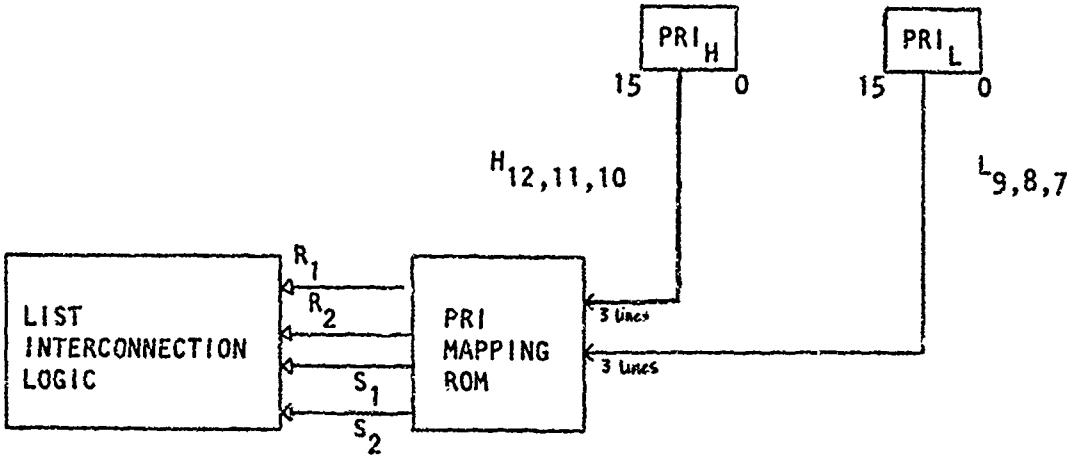
The FIFO implementation of architecture tuning of the LIST structure consists of a fixed number of FIFO integrated circuit memories of a fixed size and configuration logic. Whenever there is a change in either the minimum or the maximum PRI, the processor array passes these new values to the LIST configuration logic, which performs the following functions: (1)It maps the PRI pair (PRIH,PRIL) into the (M,N) pair based on a prestored table which was derived based on earlier research.[3] The logic for the mapping is shown in figure 5. (2)It reconfigures the LIST architecture based on the (M,N) pair generated by the mapping ROM. The configuration hardware generates the signals for proper interconnection of the FIFO ICs. Configuration will result in 2^M modules with each module consisting of one or more ICs. A logical path is created between ICs when there is more than one IC per module. For example, a reconfiguration with 16 ICs which requires a reduction in modules from 16 to 8 will cause the output from all even ICs to become input to the next IC. This will double the size of each module while halving the number of modules. (3) LIST configuration logic also uses the (M,N) pair to determine the proper module for loading and unloading data from the LIST. The NTOA is decoded to load data into the proper module of the LIST, and the real-time clock is used to unload data from the correct module of the LIST into the CAM. The low order N bits of the time fields are dropped, and the next higher order M bits are decoded into a module number.

As an example, consider a PRIH of 7900 and a PRIL of 500. The maximum value of N is 8 [$2^N=2^8=256 < 500=$PRIL]; the minimum value of N+M is 13 [$2^{(M+N)}=2^{13}=8192 > 7900=$PRIH]. M=5. So the low order 8 bits and the high order 3 bits of the NTOA of an emitter would be ignored and the middle 5 bits would be decoded into the module (from 1 to 32) into which the emitter's data should be loaded.

C.1.b. RAM Implementation

The main drawback to FIFO implementation is the large amount of hardware required to construct the LIST. The alternative to FIFO implementation of reconfiguration is a RAM memory. This implementation increases the overhead required because of the necessary use of pointers to sections of the RAM in order to construct the independent modules of the LIST. Pointers must be updated with each data word loaded or unloaded.

FIFO simulation using a RAM requires two pointers for each module: a LIST load pointer for the top of the stack (last-in) and a LIST unload pointer for the bottom of the stack (first-in). The total number of pointers to the LIST is twice



R_1	R_2		S_1	S_2	
0	0	M= 6 (64 modules)	0	0	N= 6
0	1	M= 5	0	1	N= 7
1	0	M= 4	1	0	N= 8
			1	1	N= 9

Figure 5. PRI's Mapping ROM

the number of LIST modules. In order to reduce the number of register pointers outside of the RAM memory, memory locations within the RAM itself can be used as pointers.

Reconfiguration occurs when the LOAD-CAM processor receives an interrupt from the processing array. The LOAD-CAM processor reads the new configuration status (number of modules in the LIST) and adjusts the pointers to the buffers accordingly A microprocessor with internal memory is desirable in implementing the RAM LIST because of the address computations needed to access the LIST. The internal data memory of the microprocesor is then used to keep track of pointers to the next avalable data word in sections of the LIST. Also, part of the hardware is used to map the (PRIH,PRIL) pair into the (M,N) pair configuration while the other part is used to decode the NTOA/real-time-clock into the correct module for LIST loading/unloading. LIST implementation with FIFO-ICs only requires a simple controller to perform the same tasks which needed a microprocessor in the case of FIFO-RAM.

Previous research[4] indicates that FIFO-IC reconfiguration resulted in the best performance but would be expensive to implement due to the amount of hardware required. While the FIFO-RAM approach produces additional overhead associated with updating pointers and reconfiguration of the LIST, this does degrade the system, but not significantly. FIFO-RAM implementation would be more cost effective.

D. ENVIRONMENTAL INPUTS AND PARAMETERS OF THE SYSTEM COMPONENTS

The environmental inputs and the configuration of the various components of the signal sorter system will affect its level of performance. Environmental inputs that will affect the system are the number of emitters detected in the environment at any one time, the PRIs of those emitters, the types and number of exotic emitters. Signal sorter parameters that will affect performance are the speeds of the associative processor and the array processor, and also the speed of the CAM manager. In terms of LIST configuration, parameters that affect efficiency are the [M,N] values for the number of modules and the time-slot for loading the LIST. These values and any advance load time for the CAM and the number of CAM registers will affect the performance of the whole associative processing stage of the signal sorting process. Since the efficiency of the system is dependent upon the performance of each component of the system, it is important to find the optimal configuration that will best handle any environment that could be encountered.

To be able to vary the environment and to also vary the configuration of the system, the following parameters are input to the system at run time:

ENVIRONMENT:
- -continuous wave emitters [yes/no]?
- -pulse group emitters [yes/no]?
- -nonuniform DOA change [yes/no]?
- -number of frequency hopping emitters
- -set high and low PRI [yes/accept default]?
- -total number of emitters

-separation between emitter turn-on times

SIGNAL SORTER:
-associative processing time in microseconds
-array-processor processing time in microseconds per microinstruction
-CAM manager processing time in microseconds
-number of CAM registers
-initial number of modules in the LIST (2^N)
-initial number of bits shifted (2^M)
-advance load time

Also included as inputs are 'simulation run time' and 'print increment' for outputting statistics, both in seconds.

The parameters used to evaluate the performance of the signal sorter are:

1. rate of CAM nomatches per second
2. ratio of CAM nomatch/match
3. maximum number of emitters in the assocative processor input buffer
4. maximum number of emitters in the FIFO input buffer of the array processors
5. maximum number of emitters in any bin of the LIST

The "nomatch" count in the array-processors does not include those emitters whose signals have not been identified and catalogued in the main memory files. It also does not include DOA or frequency drifts. Nomatches are those emitters whose parameters have been calculated and stored in the main memory file but whose received pulse did not match, within the prescribed limits, the parameters of any emitter in the CAM at the expected time of the arrival of the pulse. The match count represents the number of emitters matched in the CAM. The ratio of match/romatch is an indicator of how effective the filtering process of the CAM is, while the rate of nomatches per second, calculated as the difference between the total number of nomatches at the current second and the total number of nomatches at the preceding second, is an indicator of how effective CAM performance is over time, i.e., if a steady state is realizable. The AP input buffer contains output from the receiver to be matched in the CAM. The size of this buffer indicates whether the AP speed is adequate to handle the incoming data rate from the receiver system. For varying CAM processing speeds, an increase in this buffer could act as a measure of the maximum delay for CAM loading from the LIST. This will be discussed later.

A major objective of the system is to minimize the delay from the time the first pulse of an emitter is received until it is identified and stored in the system. A large number of emitters in the FIFO buffer of the array processors would result in a delay in completing their processing. This would be a factor of both CAM effectiveness and processor-array speed. Ideally, the size of this buffer should remain small over time.

The maximum size of any bin in the LIST shows the lower bound for the size of the CAM. Bin size is the maximum number of

emitters that will be loaded into the CAM during any time-slot. A large number here would increase the probability of CAM processing time causing a delay in loading an emitter in time for its NTOA. Overwriting of data items that have not yet matched is aso likely to occur if this size of any bin larger than the size of the CAM. Simulation runs indicate that for optimum performance, the bin size should not exceed half the CAM size.

These performance measures are not independent, nor are they all the performance measures that exist. These are the performance measures described in the tables of this report.

D.1. Environmental Affects

The rate of nomatches in the CAM is affected by the system configuration and varying certain input parameters; however, a percentage of those nomatches are strictly a result of the environment. Environments that contain emitters with frequencies that are equal within +1 units and are very close in direction-of-arrival can increase the rate of nomatches over time.

Emitters with DOA or frequency drifts are passed to the microprocessor array for a between-limits match of +1 units on DOA cell and freqency (frequency is scaled between 1 and 4096). If emitter-A has the same frequency+1 as emitter-B and the two are in adjacent DOA cells in memory, the microprocessor array will match and update the first location it encounters within the required limits. The microprocessor array matches only on DOA+1 and frequency+1. If A matches B on DOA and frequency, A will be updated based on the PRI of B. This will result in subsequent DOA-nomatches on emitter-A until it drifts to a nonadjacent cell (DOA drift > 1) or emitter-B drifts out of the cell adjacent to A. The former will result in a strict nomatch and entry into the memory file as a new emitter. The latter will continue to result in DOA-nomatches until the NTOA is updated to the range of the time window corresponding to the actual arrival time of the emitter.

An emitter environment generated for 300 emitters is in appendix F.3. Output includes DOA cell number, position of emitter in each DOA cell, frequency and PRI. The list is sorted by frequency. DOA's within +5 with frequencies within +1 are highlighted as a potential source for this phenomenon. The DOA drift for a several second simulation run can result in a drift of this size and larger.

From appendix F.3, consider emitter-A in DOA cell-20, position-7, with a frequency of 72 Hz and a PRI of 7623 microseconds; and emitter-B in module-18, position-4, with a frequency of 72 Hz but a PRI of 2931. If A drifts to module-19, a nomatch will occur in the CAM. The microprocessor will then search memory modules 18, 19 and 20 for a match on frequency 72+1. These modules will be searched in parallel. When a match is found in one of the modules, adjacent searches are halted. Since B is in the 4th position in module-18, it will match A before the 7th positon of module-20 is reached. As a result,

emitter-A will be updated based on a PRI of 2931 rather than 7623 and will produce more DOA-nomatches. In the worst case this could produce as many as 131 DOA-nomatches in 1 second of simulation time. The 131 nomatches corresponds to the arrival rate of emitter-A (PRI=7623).

Several such occurences in an environment will increase the number of internal arrivals (the number of arrivals to the microprocessor array due to nomatches of all types), which is an increase in the workload on the microprocessor array. This also accounts for main memory containing less than the number of emitters expec d in an environment. Instead of inserting a new emitter into the file, the microprocessor array matches the emitter with an existing one that has the DOA and frequency within the required limits.

D.1.a. Separation Time

Emitter turn-on times will also affect system performance because it alters the density of new emitters seen in the environment at any one time. In addition, if the characteristics of each emitter in the environment do not change, varying their on-times (i.e., the time at which the emitter is first seen by the signal sorter) will affect performance due to the different combination of signals that will be seen during any one time slot. It may eliminate some signal clustering; on the other hand, it may increase the occurrence of this phenomenon.

For 100 emitters and 300 emitters, runs were made with a separation time of both . 05 seconds per 100 emitters and of . 05 seconds, 100 emitters would be turned on within the first to 0.1 seconds and the third 100 emitters would be turned on within 0.1 to 0.15 seconds, and similarly for 0.15 seconds. So for an on-time separation of 0.15 seconds, all 300 emitters would not be seen until 0.45 seconds of run-time had elapsed.

The above four configurations were run long enough for all emitters in the environment to be detected. The run times were as follows:

100 emitters with	0.05 separation	-0.10 seconds
100	.15	.20
300	.05	.20
300	.15	.50

The rate of nomatches and the percentage of nomatch/match were compared. In addition, the number of internal arrivals (NIA) were examined since the rate of nomatches was at times zero for the beginning of the run. NIA includes all data that is passed to the array-processors, DOA drifts, new emitters and any other data not matched in the CAM. These values will reflect the increase in data passed to the array-processor as each emitter is detected and catalogued in main memory and the levelling off of these figures as the number of new emitters in the environment decreases and the signal sorter matches those signals it has evaluated and can now identify. Table 2 contains data from these four runs and figures 6 and 7 show graphically the rate of

Table 2. Performance as a Function of SEPARATION TIME

Run-Time Seconds	100 w .05			100 w .15			300 w .05			300 w .15		
	%N/M	RNoM	NIA	%N/M	RNoM	NIA	%N/M	RNoM	NIA	%N/M	RNoM	NIA
.01	0	0	38	0	0	16	0	0	26	0	0	2
.02	0	0	71	0	0	16	0	0	48	0	0	23
.03	0	0	68	3.1	2	23	0	0	68	0	0	19
.04	0	0	57	1.6	0	25	.95	3	78	0	0	16
.05	0	0	67	.93	0	25	.70	1	75	0	0	13
.06	0	0	14	.58	0	30	.43	0	61	0	0	27
.07	0	0	2	.39	0	28	.38	1	65	0	0	26
.08	.11	2	2	.29	0	38	.28	0	72	.22	1	25
.09	.09	0	0	.22	0	13	.21	0	46	.33	1	20
.10	.08	0	1	.17	0	17	.17	0	65	.76	5	35
.11				.14	0	20	.25	4	86	.79	2	30
.12				.12	0	15	.31	5	82	.87	3	25
.13				.10	0	19	.43	9	70	.71	0	18
.14				.13	1	26	.54	11	71	.59	0	23
.15				.15	1	14	.58	8	86	.59	2	29
.16				.13	0	14	.69	16	37	.71	5	30
.17				.12	0	0	.78	16	21	.62	0	22
.18				.11	0	0	.83	14	15	.57	1	26
.19										.54	1	27
.20										.53	2	19
.21										.50	1	21
.22										.49	2	25
.23										.46	1	25
.24										.44	1	22
.25										.41	1	13
.26										.41	2	14
.27										.41	3	27
.28										.40	1	24
.29										.40	3	29
.30										.38	1	35
.31										.36	0	38
.32										.35	2	22
.33										.38	6	23
.34										.38	3	31
.35										.41	7	32
.36										.43	7	35
.37										.48	12	28
.38										.48	3	19
.39										.49	7	41
.40										.52	10	27
.41										.58	16	37
.42										.60	11	33
.43										.62	11	38
.44										.63	9	29
.45										.67	15	39
.46										.74	25	38
.47										.73	7	11
.48										.73	7	10

%N/M = Percentage of Nomatch/Match per second
RNoM = Rate of Nomatch per second
NIA = Rate of Internal Arrivals (to array-processors) per second

o - separation time of .15 seconds/100 emitters
x - separation time of .05 seconds/100 emitters

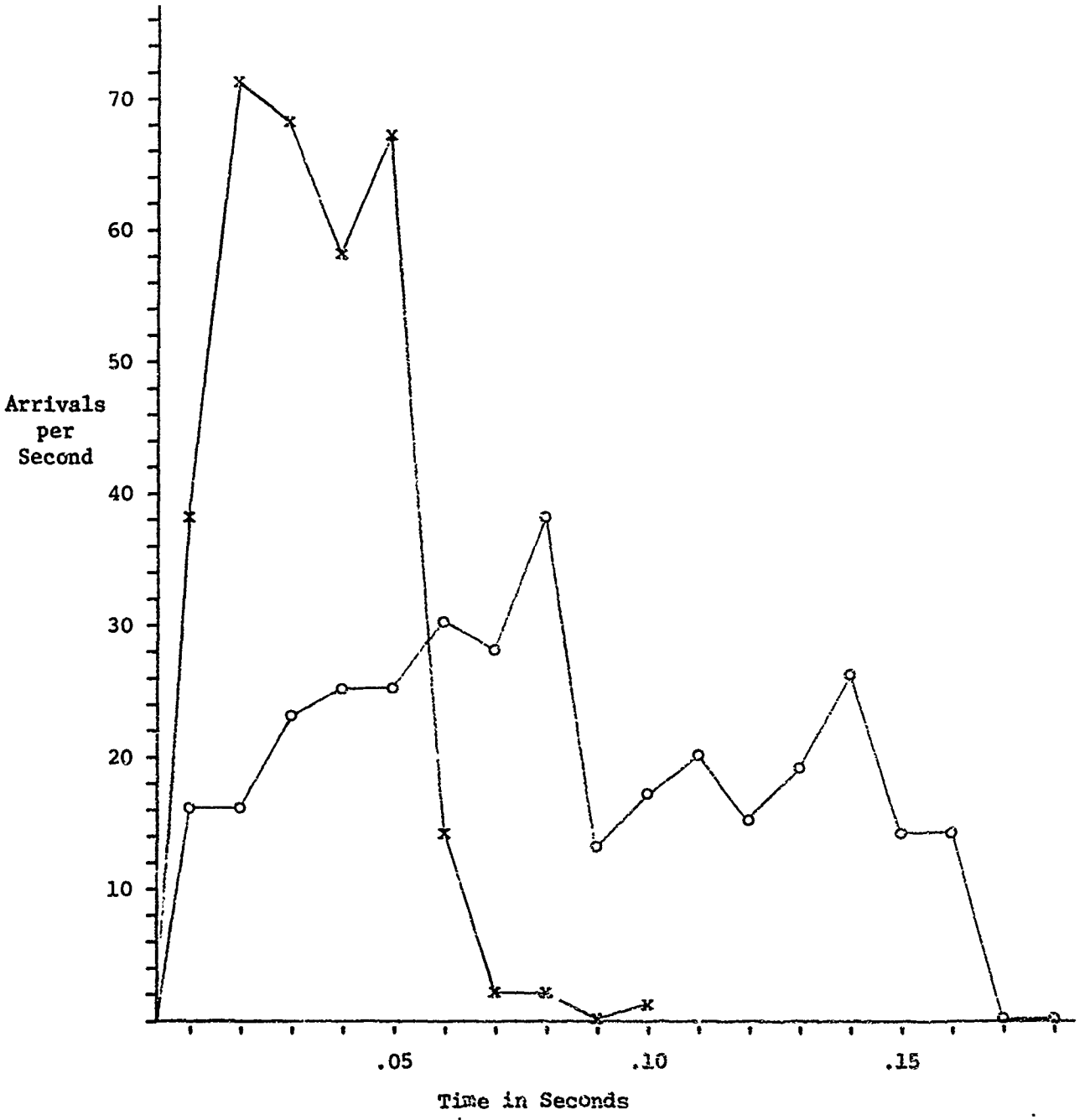


Figure 6. Rate of Internal Arrivals
For 100 Emitters

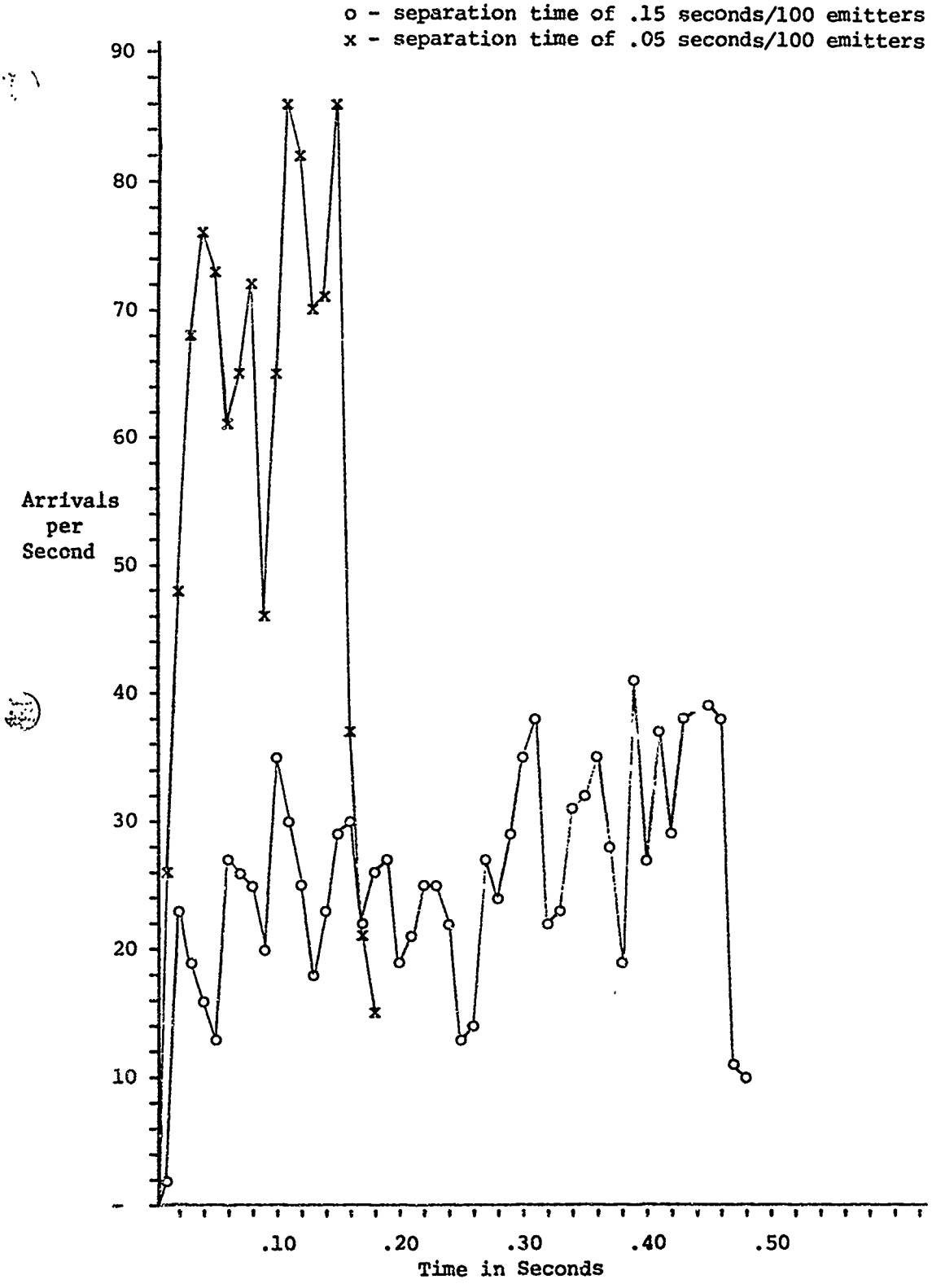


Figure 7. Rate of Internal Arrivals For 300 Emitters

internal arrivals for 100 and 300 emitters, as discussed below.

For 100 emitters with . 05 separation, the number of actual nomatches is zero up until . 08 seconds when there are two nomatches. (Recall that the nomatch count includes those arrivals which should match in the CAM under ideal predictions of arrival time and CAM loading. In other words, it does not include the nomatches from the first three arrivals of a given emitter while it is being identified by the processor array.) However, the number of internal arrivals per second shows a peak in activity at . 08 and a slightly smaller peak at that time, the number of internal arrivals is reduced significantly. The percentage of nomatch/match remains at zero until . 08 seconds when the two nomatches are produced, but this figure also decreases since there are no other nomatches during this time period. The nomatch count refers only to those emitters which should have matched in the CAM but did not for reasons other than frequency or DOA drift or new emitter.

For 100 emitters with 0.15 separation time, the first nomatches, 2, show up at . 03 seconds: there is not another one until 0.14 seconds of run-time; thus the percentage of nomatch/match can be deceiving without also looking at the rate of nomatches. The percentage of match/nomatch is a good performance measure at steady state, i.e. after all emitters are seen. The NIA rate peaks at . 08 seconds - 64 emitters have been turned on at that time - and decreases to almost a third of its peak size at 0.15 seconds, when all emitters have been turned on. After all emitters have been turned on in both runs, performance is approximately the same.

For 300 emitters with . 05 separation, all emitters will turn on by 0.15 seconds. The number of internal arrivals peaks at 0.04, 0.11 and 0.15 seconds, which is approximately the time each 100 emitters will have been added to the file. The rate of nomatches remains zero or fairly small, 5 or less, up to 0.13 seconds. The NIA rate, however, starts to decrease after 0.15 seconds. So more of the arrivals to the array-processor are nomatches rather than new emitters, which have all turned on at that point, thus NIA and rate of nomatches become approximately equal.

For 300 emitters with 0.15 separation, this does not happen until . 46 seconds of run time. The rate of nomatches remains small until about 0.35 seconds when it starts to slowly increase. Since the turn-on times are spread over a longer time, the NIA rate is much more smooth with the peaks not as pronounced as for 0.05 separation. There are slight peaks at the limit of turn-on time for each 100 emitters and usually one between these limits when at least 50 of the 100 emitters to be turned on during that time interval have been detected. For example, there is a peak at 0.16 seconds, but also one at 0.10 seconds; there is a peak at 0.31 seconds and 0.45 seconds, but also one at 0.39 seconds. After 0.45 seconds, the NIA rate starts to decrease from its peak and the nomatch rate begins to approximate it, indicating that all new emitters have been turned on.

The results of this simulation of all emitters with sign l

turn-on within such short period time is used for worst case analysis. It would correspond to a system turn-on or system reset in the midst of a dense environment or to an even denser environment with a longer separation time It represents the worst case transient analysis rather than the steady state of performance for the microprocessor-array.

D.1.b. Frequency Hoppers

Frequency hoppers are those emitters that change the carrier frequency of the pulse signal in some fashion (sometimes pulse-to-pulse) while the PRI remains the same. This type of emitter is difficult to identify. The receiver model measures the frequency and calculates the DOA based on which antenna receives the strongest part of the signal. This information is catalogued in the main memory file based on the DOA and the frequency for each detected signal is stored in the file corresponding to its DOA. Because identification is based on frequency matching, frequency hopping emitters appear as new emitters in the main memory file each time the signal appears until a pattern in the transmission is detected. The result is an increase in the workload on the microprocessors because there appears to be more emitters in the environment than are actually transmitting. Each signal is stored as a new emitter until another signal with the same frequency and DOA is received to calculate the PRI for those parameters. If the frequency is not repeated, each successive signal will be stored in the main memory file until main memory is full: in actuality, until the bin corresponding to the frequency hopper's DOA is full.

In order to determine the effect of frequency hoppers on the signal sorting system without overflowing the main memory files, a wraparound tecnhique is utilized. Whenever a bin in main memory reached its limit due to frequency hoppers, the file was modified to wrap around and start loading new emitters into the top of the bin corresponding to the DOA of the emitter. Wraparound in main memory can be detected by the maximum size of any bin reaching 48. The actual pointer bin will be to the next location that will be loaded. For those cases, the number representing total emitters in the main memory file will not reflect the fact there are 48 emitters in those bins, but will only include the size of bin indicated by the pointer. For example, if maximum size of bin 34 was 48 but the current pointer is 6, the actual number of emitters in the main memory file is the indicated total minus the pointer value plus 48 (the actual number of emitters in the bin).

An environment with 50 emitters and 2 frequency hoppers was run for 1.0 seconds. This run was compared with an environment of 50 emitters and no frequency hoppers. Both were run as a dynamic configuration with . 05 second maximum separation between on-times. One effect of the overflow in the main memory file resulting in wraparound is that the frequency hopper environment reconfigures the LIST after 0.6 seconds of run-time. Wraparound in the bin writes over emitters that were previously identified. When the signals for these emitters are detected again, they will appear as new emitters and will have to be typed and stored in

the main file again. If the frequency hopper continues to wraparound and write over the data for incoming emitters to that bin, the emitters will never be typed and the PRI will not be calculated for them. The data corresponding to the last signal will be wiped out before a new signal is detected and compared to the file. If the emitters in the wraparound bin are in the range that would produce a different configuration if these were not present, the LIST will reconfigure as if they are not present since they are not in the main memory file long enough to calculate the PRI.

Consequently, a second environment with 50 emitters and 2 frequency hoppers was run as a static configuration with a LIST configuration of (32,256), which is the configuration for the dynamic environment with no hoppers. This environment will eliminate any performance differences that are due to LIST reconfiguration.

For the dynamic frequency hopper environment (dFH) and the environment with no frequency hoppers (nFH), the rate of nomatches remains the same up until the reconfiguration of dFH at 0.7 seconds. The rate jumps from 0 to 637 in 0.1 seconds. The nFH environment remains at zero for the duration of the run. The static frequency hopper environment (sFH) is approximately the same as nFH. The slightly better performance can be attributed to the configuration remaining static throughout the run rather than reconfiguring as new emitters enter the environment. The number of nomatches in this case will reflect actual nomatches from the regular (identified) emitters since each signal from the frequency hoppers will appear as a new emitter, not as a nomatch on an existing emitter. The indicators of the effect on the system are most apparent in the number of emitters that appear to be in the environment and the ratio of the number of matches to the number of signals received. The number of external arrivals (NEA) will be the same for all runs since all three environments have the same parameters with the exception of the two frequency hoppers that only vary their frequency with each pulse. Initial values are the same.

The actual number of emitters that are in the main memory file for each run are in table 3. At 0.6 seconds for dFH, the number of emitters in the main memory file is listed at 80. However, the actual number is 141 since the overflow in bins 34 and 16 is not added in. The current sizes of those two bins are 6 and 29, respectively. These two values must be subtracted from the printed total of 80, and 48 must be added to the resulting total for each full bin to get the correct total. The computation will be 80-(6+29)+2*48, which is 141 emitters. Of course, the nFH environment shows the total number of emitters in main memory as 50.

Table 3 also shows values for the percentage of match/NEA. The figures for sFH are approximately 3% less than that for nFH. The dFH environment is equivalent to sFH until the reconfiguration at 0.7 seconds. The PRI of the frequency hoppers will determine how quickly main memory will overflow; as it is clear that the emitter in bin 34 has a highter PRI than the emitter in bin 16, both frequency hoppers. The results of the

Table 3. Performance as a Function of FREQUENCY HOPPERS

Run-Time Seconds	FH=0 (32,256)	s FH=2 (32,256)	d FH=2 (32,256)
	RATE OF NOMATCHES		
0.1	2	0	2
0.2	1	1	1
0.3	0	0	0
0.4	0	0	0
0.5	0	0	0
0.6	0	0	0
0.7	0	0	637
0.8	0	0	841
0.9	0	0	866
1.0	0	0	860
	PERCENTAGE of MATCH/NEA		
0.1	87.5	85.1	85.0
0.2	94.5	91.9	91.8
0.3	96.4	93.8	93.7
0.4	97.3	94.6	94.6
0.5	97.9	95.1	95.1
0.6	98.2	95.4	95.4
0.7	98.5	95.7	90.0
0.8	98.6	95.8	84.3
0.9	98.8	96.0	79.8
1.0	98.9	96.1	76.3
	NUMBER OF EMITTERS IN MEMORY FILE		
0.1	50	88	88
0.2	50	117	117
0.3	50	130	130
0.4	50	140	140
0.5	50	139	139
0.6	50	141	141
0.7	50	141	141
0.8	50	146	147
0.9	50	157	159
1.0	50	170	174

NEA = number of external arrivals (from Receiver)

sFH environment are in appendix F.6.a.

The same environment was run as a static environment of 50 emitters and 2 frequency hoppers with no wraparound since, ideally, new emitters would not be added to a full module or when a module is full, a search to detect frequency agile emitters should be executed. When the memory module is full, new emitters and DOA-drifting emitters are not added to the file. This means they will not be loaded into the CAM for match, but will always appear as new emitters. They will be passed to the microprocessor-array for entry into the file but will not be loaded. Thus, the only indication of this extra arrival activity will be the increase in the number of arrivals to the microprocessor-array (NIA) and the match count. The difference in NIA for sFH and the non-wraparound memory environment is less than 2% at the end of run-time; however, simulation run-time was only 1.0 seconds. For example, at 1.0 seconds for sFH, the match count is 15708 and NIA is 641 compared to 15694 and 655 for the static memory case (no wraparound). One of the main differences in the two runs is the increase in the nomatch count at 0.6 seconds for the static memory case due to a DOA drift from memory module 17 to memory module 16 at the end of 0.5 seconds. Since module 16 is full, the emitter data is transferred out of module 17 but cannot be loaded into module 16 The results of the static memory case are in appendix F.6.b.

Obviously, frequency hoppers in the environment will degrade the system considerably if there is no provision for identification of these signals. As seen above, this results in memory overflow, and if reconfiguration takes place, degradation of performance in matching incoming signals since the system will not adequately catalog the environment.

D.2. LIST Configuration

Reconfiguration of the LIST dynamically according to maximum and minimum PRI's should produce the best performance, based on research cited earlier. The PRI range for the simulation runs made for this report is from 500 to 7900 microseconds. Expected values for the number of bits shifted, N (shown as a displacement of 2^N), and the number of bits in the time-slot-window, M (shown as 2^M, representing the number of bins in the LIST), is 256 and 32, respectively.

This (M,N) pair was derived based on a maximum value of M being 7: that is, a maximum of 128 modules in the LIST are allowed. This limit was placed arbitrarily to limit the hardware cost of FIFO-ICs. This limit and the resulting (M,N) pair for the above environment worked well for a total number of emitters of 100.

An environment of 300 emitters was run with both static and dynamic configuration using 256 and 32 as the input parameters for the LIST with approximately equal results, as was expected. This same environment was run as a static configuration with the number of bins for the LIST larger than the requirement, with the values of M and N still satisfying the restrictions (1) $N < [\log_2$

PRIL] and (2) N+M > [$\log_2$ PRIH]+1. This required a reduction in the number of bits shifted: the offset for the time-window.

The number of bins was increased to 128. the time-window-offset was reduced to 64. Using the percentage of NOMATCH/MATCH in the CAM as a performance measure. this value decreased by a factor of 17 from the (32,256) configuration. For the (128.64) configuration at 8.0 seconds. the percentage of nomatch/match was 1.55 compared to 26.2 for the (32.256) configuration. See table 4 for results. However, when this LIST configuration was used for 100 emitters, performance was degraded.

One explanation for the improved performance is that the smaller bin size due to the larger number of bins in the LIST results in fewer nomatches in the CAM due to avoiding overwrite. The CAM is loaded from a bin in the LIST based on a time-window. If the number of items in any one bin is large. loading the CAM may overwrite data that has been updated and the NTOA of that emitter has not passed. This is more likely to happen when the number of emitters in any one bin is larger than or equal to size of the CAM. The smaller offset to the time window will require more CAM loads, but it increases the likelihood of a data item being matched before another data item is loaded into that register. The (128.64) configuration resulted in the bin size for 300 emitters being reduced to the range of bin sizes of the (32,256) configuration for 100 emitters.

For improved performance, a large number of modules in the LIST should be used whenever the expected number of emitters in the environment is large. For dynamic reconfiguration to be effective, the maximum value of M (i.e., M-max) should be a function of the expected number of emitters in the environment. Monitoring both the PRI and the maximum size of the LIST bins. which is. in turn. dependent on the number of emitters (NE), should result in optimum performance of the system. In other words, reconfigure the LIST according to PRI and LIST size (or NE). with the limit being the size of the CAM.

The optimum configuration values that are used in the simulation. as discussed in section C.1., were based on a maximum environment of 100 emitters. The optimum values for larger environments were not calculated. Still, the larger value for M satisfies the condition: $2^M \geq$ maximum PRI.

D.3. Associative Processor (AP)

Data is loaded into the CAM during the same time-slot as the emitter's NTOA: i.e., an emitter is loaded into the CAM in advance of its NTOA by an amount of time less than or equal to the size of the time slot minus the delay in the LIST. However, the CAM is loaded from the LIST only when it is not processing data from the receiver. The NTOA of an emitter may require a CAM load during this period of processing. If the processing time for data from the receiver results in current time greater than the NTOA of emitters in the LIST, this will result in nomatches in the CAM due to late loading of the CAM or not loading a data

Table 4. PERCENTAGE OF NOMATCH/MATCH

Number of Emitters	Run-Time Seconds	(32,256)	(32,256) MPT=.01	(32,256) CPT=0.5	(128,64) MPT=.01
100	1	.099	.096	.016	.609
	2	.107	.104	.022	.597
	3	.119	.116	.031	.559
	4	.129	.125	.031	.548
	5	.124	.121	.031	.535
	6	.119	.116	.031	.529
	7	.118	.115	.027	.517
	8	.113	.110	.027	.516
	9	.115	.112	.030	.516
	10	.127	.125	.038	.515
	11	.129	.126	.038	.517
	12	.131	.128	.038	.516
		(128,64)	(128,64) MPT=.01	(128,64) CPT=0.5	(32,256)
300	1	1.43	1.35	.197	24.7
	2	1.44	1.36	.193	25.5
	3	1.46	1.39	.194	25.8
	4	1.49	1.42	.204	25.9
	5	1.51	1.44	.222	26.0
	6	1.52	1.45	.231	26.1
	7	1.53	1.47	.246	26.1
	8	1.55	1.48	.251	26.2

LIST SIZES Minimum*, Maximum

Seconds Run-Time	100 Emitters (128,64) , MPT=.01	300 Emitters (32,256)
1	5,10	39,40
2	6,10	40,50
3	7,10	40,50
4	7,10	41,50
5	7,11	42,50
6	7,11	42,50
7	7,11	42,50
8	7,11	43,50
9	7,11	
10	7,11	
11	7,11	
12	7,11	

*smallest maximum: Also see Table 6 for LIST sizes

item from that module in the LIST because the time window has advanced beyond the time slot of that module.

In order to determine the degree to which CAM processing time degrades the performance of the signal sorter, a simulation run with CAM-manager processing time (CPT) set to .05 was evaluated, with the following other input parameters:

number of emitters = 300
maximum on-time separation/100 emitters = .05 seconds
simulation run-time = 1.0 seconds

The results of this run were compared with a run in which all the parameters except CPT were identical. CAM processing time in the comparison run was set at 1.0 microseconds, which is in the range of current technology. See table 5 for the results from each run.

The number of nomatches in the CAM dropped by a factor of approximately 100 from 1464 to 15. The buffer to the micro-processor array and the input buffer to the LIST were reduced by a factor of 2. Such a dramatic reduction in nomatches indicates that either a reduction in CAM processing time or a change in loading time may significantly improve the performance of the signal sorter.

Consequently, the same environment parameters were used in another run with a CAM processing rate of 0.5 microseconds. This rate is also within the range of current technology. Table 5 also includes results for 100 emitters. The 50% reduction in CPT produced an improvement in performance over the 1.0 microsecond processing time by a factor of 6 and 7 for 100 and 300 emitters, respectively.

Since larger AP processing time, in some cases, results in late loading of the CAM, the effect of loading a data item into the CAM in advance of the NTOA also requires evaluation. This will be discussed in section D.4.

As long as the size of any bin in the LIST remains smaller than the size of the CAM, there is no possibility of an emitters' parameters being overwritten before the signal is matched in the CAM. This does not, however, alleviate the problem of late loading or no loading of an emitter before the NTOA of the emitters signal. (Loading the CAM has a lower priority than processing the incoming data; therefore, loading is performed between arrivals.) The problem of late loading or no loading has a greater effect on the emitters with small PRI. A modified CAM structure which treats these small PRI emitters differently from other emitters was analyzed and is presented below. A small section of the CAM was aside for emitters with small PRIs to determine the effect on performance. A permanent copy of the emitter was kept in the CAM to avoid the delay of updating and reloading the emitter. This would result in a miss if it is larger than the PRI of the emitter. The reserved section of the CAM has a copy of those emitters with PRI less than a specified minimum. These registers will not be overwritten but will hold the DOA and frequency of the arrivals with the smallest PRI on a

Table 5. CAM Processing Time (CPT) in Microseconds

No. Emitters (LIST Config)	Advance Load Time (ALT)	Number of Nomatches		
		CPT=.05	CPT=0.5	CPT=1.0
100 (32,256)	0	--	6	36
	4 micro-secs	--	7	16
	8 micro-secs	--	10	10
	16 micro-secs	--	31	27
300 (128,64)	0	15	--	--
	0	--	204	1464
	4	--	46	437
	8	--	48	118
	16	--	92	--
	32	--	206	--

The following input parameters were constant in all cases:

max separation between on-times/100 emitters	.05 seconds
associative processing time in microseconds	1.0
array-processing time per microinstruction	0.1 microseconds
number of CAM registers	24
run time	1.0 seconds

first-arrival basis. When this section is full, all other emitters will be loaded into the remainder of the CAM and updated in the standard manner.

From one to four words of the 24 CAM registers were reserved for emitters with a PRI of 800 microseconds and less. For the particular environment run, there were nine emitters with PRI less than 1000, five with PRI less than 800. For a maximum PRI of 800 microseconds, setting aside three CAM registers resulted in the best performance over that of the no-reserve CAM. This is a better performance than the reserve of 5 registers, which takes care of all emitters with PRI less than 800. For results of these runs, see table 6.

For all the test runs with the reserve-CAM, the environment was the same. Consequently, the only performance measure that changed was the number of nomatches. All other output parameters remained the same. Table 7 contains the order of arrival and the PRIs of emitters with PRI less than 1000.

A one-word reserve with minimum PRI of 800 [PRI=642] resulted in one less nomatch than the one-word reserve with minimum PRI of 600 [PRI=500]. While the opposite is expected: that the emitter with the smallest PRI would produce the better performance, the difference is very slight. Also, for nine reserved words and a minimum PRI of 1000, the performance for . 2 second run is the same as that for no reserved words.

It appears that performance with a reserved CAM section not only depends on the number of small PRI emitters in the environment but also on the combination of signals that are seen at any one time. Over the long run, a small reserved-CAM section may improve performance: provided that the maximum size of any bin in the LIST does not exceed the number of CAM registers, excluding the number of reserve-CAM words. A reserved section of the CAM will increase the amount of time required for updating the CAM since it will be necessary to do this for two sections of the CAM: the reserved section must be checked to be sure all emitters are still in the environment and if the DOA or frequency has drifted. The small and uncertain gains in performance do not warrant a change at this time when other improvements may produce larger gains.

D.4. Advance Load Time (ALT)

The Associative-processor input buffer holds data items that are passed to the CAM from the receiver. The maximum number of emitters in this buffer at any one time can act as an indicator of the maximum amount of time a CAM load may be delayed: the product of the size and the CPT. This figure does not take into account other factors such as fetch time for a new data item from the buffer, but it can act as a barometer for determining which range of value for advance load time (ALT) will result in optimum performance.

For a 1.0 second simulation run with 100 emitters, the maximum AP buffer size was 3. This environment was run, varying

Table 6. Reserved CAM

Run-Time Seconds	# CAM Words Reserved	Minimum PRI	Number of Nomatches
0.2	0	---	7
0.2	1	800	5
0.2	2	800	3
0.2	3	800	3
0.2	4	800	3
0.2	5	800	3
0.2	9	1000	7
1.0	0	---	36
1.0	1	800	33
1.0	2	800	32
1.0	3	800	26
1.0	4	800	27
1.0	5	800	28
1.0	1	600	34

Table 7. PRIs for Emitters with PRI < 1000 by Order of Arrival

Order of Arrival	PRI
1	642
2	920
3	922
4	659
5	728
6	922
7	618
8	500
9	943

Input parameters for data in tables 6 and 7

100 emitters
.05 separation between on-times/100 emitters in seconds
1.0 Associative Processing Time in microseconds
0.1 array-processing time in microseconds/microinstruction
1.0 CAM processing time in microseconds
24 CAM registers

the ALT of an item into the CAM by 4, 8, and 16 microseconds. The best performance was for an ALT of 8 microseconds. However, reducing the CPT with no advance load gave slightly better results. Of course, the probability of overwriting data in the CAM is increased with the ALT factor, which may explain the slight decrease in performance. An ALT of 8 microseconds produced the same results with CPT both 1.0 and 0.5 microseconds. Reducing the CPT to 0.5 with an ALT of 4 microseconds brought the nomatch count to within 1 unit of the configuration with no advance load. The results are shown in table 5.

For 300 emitters, the combination of ALT and 0.5 CPT produced the best results. The maximum AP buffer was 4. With CPT set at 0.5, an ALT of 4 produces a slight improvement (4%) over an ALT of 8. The smaller ALT may overcome the effect of the CPT delay without causing much CAM overwrite. Because of the larger number of emitters in the environment and clustering of signals at times, the probability of overwrite increases.

D.5. Variable Environments

Simulation runs with the same number of emitters in the environment generate emitters with the same environmental characteristics. The random values generated will be the same for any run. What will be different will be the parameters to which the values are assigned. Since each emitter has the same number of parameters associated with it, if the number of emitters in the environment remains the same, each emitter will be reassigned the same value it had in a previous run. Varying separation time will result in a different on-time for each emitter but the same value of DOA, frequency and PRI associated with the emitter in a previous run will be assigned in a new run except the TOAs will differ by a constant. However, varying the number of emitters in an environment by even a small factor changes the environment generated because the random numbers will remain in the same sequence but the parameter to which each number is assigned will be different by an offset related to the difference in the number of emitters in one environment over another. As a result, a value (between zero and one) used to calculate the PRI of an emitter for a 100-emitter environment may now be used in a calculation for frequency in a 300 emitter environment. Varying the number of emitters by even a small amount will result in a different environment. This property of the simulation program could be used to generate different environments by simply varying the number of emitters even by one. This will not consititute too much change in the overall system workload but will change the individual parameters of the emitters in the environment.

As an example, a run was made with all other input parameters equal and with number of emitters at 100 and at 102. There was a difference in the environment generated and a slight difference in the performance of the system. For the 100-emitter environment, the minimum frequency for an emitter was 5 with a corresponding PRI of 7565, while for the 102-emitter environment, the minimum frequency was 6 with a corresponding PRI of 6025. Minimum PRI for both runs was 500 with a corresponding frequency

of 3017 for 100-emitters and 2844 for 102-emitters. A table of the environment generated for both these runs is in appendix F.4, and includes DOA, frequency and PRI. The tables are sorted by ascending PRI and also by ascending frequency.

The 100-emitter environment outperformed the 102-emitter environment slightly, with nomatch/match rate being 0.099% to 0.112%. Of course, for the 102-emitter environment, the number of external arrivals to the signal sorter is greater for the same amount of processing time since the separation time remained constant. This, of course, increases the workload on the system slightly, as can be seen in the difference in the number of external arrivals to the system. For 100, it is 36853, and 36940 for 102: a difference of 87 signals within one second. These results indicate that the performance of the system does not vary much with variations in the individual parameters of the emitters.

D.6. Long-Run Analysis

Simulation runs were made for 100 and 300 emitters to determine the performance of the system over longer periods of time: particularly with regard to LIST sizes, buffer sizes and rate of nomatches over the length of the run. For all runs, unless otherwise noted, the following parameters were used as inputs:

Maximum on-time separation/100 emitters	.05 seconds
AP time in microseconds	1.
Array-processing time/microinstruction in microseconds	0.1 (MPT)
CAM-manager processing time in microseconds	1. (CPT)
Number of CAM registers	24

Maximum on-time separation is the amount of time before the the total number of emitters in the environment are turned on. A hundred emitters are turned on in the first period. If the the number of emitters in the environment exceeds 100, the next 100 emitters will be turned on in the next period. Thus, for 300 emitters with separation=.05, 100 emitters will be turned on the first .05 microseconds, the second 100 emitters will be turned on between .05 and 0.1 microseconds and the third 100 emitters will be turned on between 0.1 and 0.15 microseconds. The standard LIST configuration for 100 emitters will be the dynamic case with an initial number-of-modules/time-window-offset of (32,256). For 300 emitters, the environment will be run as the static case with a LIST configuration of (128,64). These values were chosen since they produced a LIST size that did not exceed the size of the CAM. This was necessary since the current configuration routine was not optimized for an environment in which the number of emitters was larger than 100. An upper bound of 64 on the maximum number of modules in the LIST was self-imposed. Simulation results have shown that the optimum number of modules in the LIST is below this maximum for 100 emitters. This is not true for 300 emitters.

The (128,64) configuration for 300 emitters still satisfies

the restrictions on the (M,N) pair as discussed in section D.2 - LIST Reconfiguration. The current (128,64) configuration is approximately equal to the case in which the LIST would dynamically reach that configuration: the difference being the overhead required for the system to adjust to each new emittr as it enters the environment in order to reach th (128,64) configuration. This would happen if the restriction on the number of LIST bins in the current configuration routine are removed and if, in addition to configuring based on the maximum and minimum PRI, the size of the LIST bins is also considered. Configuration based on PRI distributes the emitters over the number of bins in the LIST while the check on the LIST bin sizes produces a compatibility in CAM size and LIST bin size: i.e., that LIST bin size does not exceed te number of CAM registers.

Six configurations were run with time restrictions based on overflow in main memory. For 300 emitters, time was limited to eight seconds, and for 100 emitters, time was limited to 12 seconds. The six configurations chosen for the longer runs were those that produced the best results in prior analyses. Details for each are as follows:

300 emitters - standard
- MPT= 0.01
- CPT= 0.5

100 emitters - standard
- MPT= 0.01
- CPT= 0.5

D.6.a. ENVIRONMENT MODEL: SPECIAL PROBLEMS

The environment model used in this simulation presented a special problem for nonstationary environments --environments in which the signal sorter and/or emitters were mobile. Because of a gremlin, as yet undiscovered, in the routine that updates environment parameters, there is a DOA drift biased toward the first antenna in bin-3. As result, the third bin of the main memory module for storing the parameters of emitters with a DOA of three fills up more quickly than others and results in an error message "MPPR FILE FULL" and system halt after a relatively short run time (8 seconds simulation time for 300 emitters, 20 for 100). To determine the degree to which this problem due to the mobility of the environment, 300 emitters were run as a stationary environment; this is not a practical case - it can never happen on an airborne platform.

When a 300-emitter environment was run with the same input parameters as the one mentioned but with a stationary platform and no moving emitters, this overflow of the array-processor did not occur. Of course, since there were no moving emitters, there were no DOA changes as a result of movement from one DOA cell to another. There is a DOA drift that is a result of a frequency drift and can be traced to a problem cited earlier, that of two emitters having frequency and DOA both within +1 of each other. This resulted in the emitter being moved out of DOA cell 48 and updating the emitter in cell 47, the adjacent cell that matched its parameters. This same emitter may account for the other

DOA/frequency drifts that occur after 10 seconds of simulation run-time. This configuration was run for 20 seconds with no overflow.

Of course, this environment of 300 emitters is different from the standard environment (i.e., a 300-emitter environment with moving emitters and platform). The difference is a result of the number of calculations using the random number generator, as discussed in section D.5. In this case, the sequence of random numbers will be offset by the number of calculations required to produce the initial values and updates for moving emitters. This environment will be different; in addition, performance could also be slightly affected.

Compared to the active run (moving emitters and platform) nomatch/match percentages and the rate of nomatches is better on the order of approximately 200 and 300 percent, respectively. The rate of both measures, however, continues to increase over time. The results of this run are in appendix F.5. See table 8 for percentage of nomatch/match and rate of nomatches.

In order to increase run-time without a file overflow, the the maximum size of a bin in the main memory file was increased from 24 to 48. This, consequently, increased the maximum delay through the system of a data word (MAX in the statistical package). A larger memory must be searched for each nomatch. Hence, delay could not be used as a performance measure.

The effect on nomatch count cannot be predicted. However, the probability of nomatches due to environment characteristics increases, as discussed in section D.1. One simulation run of 300 emitters halted after a run of 14 seconds with 304 emitters in the main memory file. Each main memory cell had been increased to a maximum of 48.

The large number of emitters in the same DOA cell increases the search time in that module. In order to counteract any effect this might have on the data collected, data from runs longer than one second simulation time will not include data generated after the size of bin three exceeds the maximum in other bins by a factor of two (2).

For 100 emitters, the maximum run-time will be approximately twelve seconds and a maximum of approximately six seconds for 300 emitters. These run times were determined empirically.

D.6.b. LIST SIZES

Outputs of the statistical package which are useful in evaluating the effectiveness of the Associative-processing stage of the signal sorting process are the maximum LIST sizes at any one point in time and the actual LIST size at that point. The maximum LIST size is taken as the largest number of emitters in any one bin of the LIST. The value will not be the same in all bins; but over time, the other bins will reach this maximum as a new NTOA is generated for each emitter in the LIST. Assuming this value is the maximum number of emitters that need to be

Table 8. 300 Emitters - Stationary Run Performance

Runtime in Seconds	Percentage of Nomatch/Match	Rate of Nomatch/Second
1	.59	566
2	.60	648
3	.60	628
4	.60	624
5	.61	686
6	.62	67[illegible]
7	.62	680
8	.62	658
9	.63	700
10	.65	710
11	.63	687
12	.63	677
13	.64	722
14	.64	665
15	.64	689
16	.64	687
17	.65	847
18	.66	840
19	.66	839
20	.67	843

No moving emitters, stationary platform
Other input parameters:
.05 Separation Time
1.0 Assoc Proc Time
0.1 Mpp Proc Time/Microinstr
0.5 Cam Proc Time
24 Cam Registers
128,64 LIST Config

loaded into the CAM during any one time slot, if this number is larger than the CAM size, nomatches will result from not being able to load all the data items into the CAM in time for a match.

For the environment with 300 emitters, the maximum value in any bin of the LIST was 22, which is slightly less than the number of CAM registers (24). This peak was reached at one second and did not increase over the eight second simulation run. However, the maximum in all bins did increse as emitters moved into different time slots for their NTOA, as was expected. The smallest was 12 and increased to 14 by eight seconds. See table 9 for LIST sizes for these runs: minimum and maximum values. For 100 emitters , the maximum reaches 22 at 3 seconds of simulation run-time and remains at that value until 12 seconds, when it increases to 23. This increase may be attributed to clustering of signals at that point. Only one bin in the LIST was incremented. Judging from the length of time the maximum remained at 22, maximum bin size should remain stable over time for a constant environment. It should also be kept in mind that this is the maximum value reached in a particular bin for the length of the run and does not reflect the average size of a bin for the length of a run.

D.6.c. BUFFER SIZES

The statistical package includes output for the maximum size of:

1. the input buffer to the CAM (data from receiver)
2. the input buffer to the array-processors (MPPB)
3. the input buffer to the LIST (MLCAMB)

For all six runs, the maximum sizes of these buffers do not increase significantly over the length of the run, as shown in table 10. Increases in any buffer size are directly proportional to a large increase in the rate of nomatches at that point in time (table 11). For example, for emitters with CPT set at 0.5, the increase in MPPB and MLCAMB at five seconds is reflective of the increase in the rate of nomatches at five seconds. The data stream going into the array-processor's input buffer and the data stream out of the array-processors into the LIST is determined by the effectiveness of the CAM match process.

For 100 emitters, the increase in the CAM-buffer at ten seconds may be indicative of an increase in the number of signals coming from the receiver due to a clustering of emitters with very close NTOAs. As a result, there is a significant increase in the rate of nomatches at that time from approximately 50 to 90 for the dynamic cases.

The fairly constant size of these buffers indicates that the speeds of the CAM-match, array-processor update, and LIST loading are sufficient to keep up with the data rate of the environment. Decreasing the CPT had no effect on the CAM input buffer; hence, CPT at 1. microsecond is adequate to handle the data rate. The reduction apparently was only necessary for adequate time to load the CAM from the LIST. Decreasing MPT resulted in a reduction in

Table 9. LIST Sizes: Minimum* Maximum

No. Emitters (LIST Config.)	Run-Time Seconds	Standard	MPT=.01	CPT=0.5
100 (32,256)	1	15,21	15,20	15,20
	2	16,21	16,21	16,21
	3	17,22	17,22	17,22
	4	"	"	"
	5	"	"	"
	6	18,22	18,22	18,22
	7	"	"	"
	8	"	"	"
	9	"	"	"
	10	"	"	"
	[illegible]	"	"	"
	12	18,23	18,23	18.23
300 (128,64)	1	12,22	12,22	12,22
	2	13,22	13,22	13,22
	3	14,22	14,22	14,22
	4	"	"	"
	5	"	"	"
	6	"	"	"
	7	"	"	"
	8	"	"	"

*Minimum represents smallest maximum

Table 10. BUFFER SIZES: CAM/MPPB/MLCAMB

Number of Emitters	Run-Time Seconds	Standard	MPT=.01	CPT=0.5
100 (32,256)	1	3, 3, 2	3, 1, 1	3, 4, 3
	2	"	3, 1, 2	"
	3	"	"	"
	4	3, 3, 3	"	"
	5	"	"	"
	6	"	"	"
	7	"	"	"
	8	"	"	"
	9	"	"	"
	10	4, 3, 3	4, 1, 2	4, 4, 3
	11	"	"	"
	12	"	"	"
300 (128,64)	1	4, 6, 7	4, 2, 3	4, 3, 4
	2	"	5, 2, 3	5, 3, 4
	3	"	"	"
	4	"	"	"
	5	"	"	5, 5, 6
	6	"	"	"
	7	"	5, 7, 8	"
	8	"	"	"

Table 11. RATE OF NOMATCHES PER SECOND

Number of Emitters	Run-Time Seconds	Standard	MPT=.01	CPT=0.5
100 (32,256)	1	34	35	6
	2	43	42	10
	3	54	52	19
	4	59	59	12
	5	40	39	11
	6	35	34	11
	7	42	41	3
	8	30	29	10
	9	48	48	19
	10	90	88	41
	11	55	53	15
	12	55	54	16
300 (128,64)	1	1666	1568	230
	2	1620	1533	216
	3	1682	1610	222
	4	1768	1699	265
	5	1758	1685	329
	6	1770	1687	312
	7	1801	1750	380
	8	1822	1759	326

the input buffer, as was expected, since the processing time for each data item was reduced by a factor of ten (table 10). This did not result in a significant reduction in the rate of nomatches over the standard processing time. The 0.1 microsecond processing time is apparently sufficient for the two environments.

D.6.d. Rate of Nomatches

The rate of nomatches per second can act as a barometer for determining if a steady state of the system for a particular environment has been reached; i.e., there is no steady increase in this rate. For the 100 emitter environment, the nomatch-rate fluctuates over the length of the run, but it does not appear to be increasing. There is a dramatic jump in the nomatch-rate at 10 seconds, which may be a result of a clustering of signals at that point or an anomaly in the environment generator. However, all other values remain within a constant range for the duration of the run; i.e., a steady state has been reached. For 300 emitters with CPT=0.5, the nomatch-rate seems to be stabilizing at approximately 5 seconds of run-time. For the two other cases, the rate is increasing. It will require longer run-times in all three cases to determine if a steady state has been reached. This was not possible with the run-time restrictions (see section D.6.a.).

The parameter that produced the most significant improvement in the rate of nomatches was the reduction in CPT, as was discussed in section D.3. This was true for both environments. This did result in an increase in the input buffer to the array processors but this could be remedied by also increasing the array-processor speed. It is found that increasing the array-processor speed will not significantly decrease the rate of nomatches, but it will decrease the size of the buffer since data items are being processed more quickly.

E. Conclusions

Since the primary purpose of the research discussed in this report was to study the applicability of dynamic tuning to a particular signal sorting system, the emphasis of the work done has been on the Associative Processing stage of the signal sorter design. The Array Processors also come into play since all new data and all unmatched data are passed to this stage. The parameter which produces the best improvement in performance is reduction of the CAM processing time, which is the amount of time required to match a received signal to a data item in the CAM. This operation takes precedence over loading the CAM with new data or updated data; consequently, data may not get loaded into the CAM in time for a match on the incoming signal. Making the CAM processing time as small as current technology will allow, will decrease the amount of time that signals are missed only because there was no time during the gap between CAM searches. In fact, of all parameters tested, CAM Processing has the greatest effect on the system, as was seen by reducing this speed to .05 microseconds, which is not in the range of current

technology. However, reducing the speed from 0.1 microseconds to 0.5 microseconds resulted in an improvement of performance by at least a factor of six (6) as measured by the match/nomatch rate. So the greatest handicap to the system is getting data loaded into the CAM in time for the NTOA of a signal.

Some consideration was given to loading data items into the CAM in advance of the NTOA by some constant time value in an effort to offset late loading or no loading of the CAM due to CAM search time. The results, in section D.4, appeared to be too dependent on the environment to produce consistent results over the long run. Also, better results were obtained by reducing the CAM processing time.

Another aspect of the Associative Processing stage is the reconfiguration of the LIST based on the environment only in terms of PRI. Since the optimal table for reconfiguration was based on a 100 emitters environment, dynamic configuration for a more dense environment does not produce a satisfactory performance level when the number of expected arrivals in any one bin of the LIST exceeds the size of the CAM. Of course, this would only require a minor change and would take a slighlty longer time for reconfiguration due to the amount of time to check the size of the LIST bins. It would require an increase in the maximum number of bins required from 64 (2^6) to 128 (2^7). The maximum number of bins in the current configuration tables is 64.

With regard to FIFO and RAM implementation of the LIST, shortly after the investigation of the FIFO-RAM (section C.1.b), Signetics announced the development of a controller chip which converts RAM to FIFO buffer memory. A single controller chip can handle up to 4096 buffer words. Word width is defined by the user. Consequently, FIFO depth could be 64, 256, 1024 or 4096.[5]

The Array Processors handle data not matched in the CAM by doing a between limits search on the DOA and frequencies of emitters in its file. This presents a problems only when two emitters in adjacent DOA bins have frequencies within +one of each other because the emitter may be matched on the wrong signal, in which case it will be updated based on the wrong PRI. Unfortunately, there is no simple solution to this this problem. The main memory file can be searched for other signals that satisfy the match. If there is more than one, the signal should not be updated but held until another unmatched signal with the same parameters is received. Then the signal can be matched to the main memory file based on DOA, frequency and PRI. The frequency and the probability of these nomatches may not warrant the extra time required for the memory search and the extra hardware required for the match.

Exotic emitters such as frequency hoppers will greatly degrade the system configuration presented here unless there is some algorithm for detecting these signals. Otherwise, each signal will appear to be a new signal and get loaded into main memory each time the signal appears. This will result in filling up a main memory module and prevent loading of other signals, resulting in nomatches and perhaps system halt. This will also

result in reduced efficiency in processing other emitters that are in that module. In the case where the main memory file is allowed to wraparound, tne environment PRIs may not be adequately detected and this will affect LIST configuration.

Overall, the particular signal sorter design studied here exhibits a steady performance for a 100 emitter environment. For 300 emitters, longer runs are required to determine if a steady state has been reached. With the changes discussed above, the performance of the system can be improved, particularly with a reduction in CAM Processing time. Further study can be done on the performance of the array-processors and on performance of denser environments over longer runs.

REFERENCES

1. R.H. Evans. Architectural Considerations in Signal Sorter Design, dissertation at George Washingtion University. Washington. DC. 1975. pp. 41-73.

2. A.M. Abd-Alla, et al. Architecture Tuning of a Real-Time Signal Sorter in a Dense Environment. Naval Research Laboratory Report 8495, Washington, DC, Sept. 81. pp 9-15.

3. Ibid.

4. Ibid.. p. 22.

5. Jeff Seltzer. "Organize RAMS as FIFO Buffers with an LSI Controller Chip", Electronic Design, Sept. 30, 1981.

APPENDICES

Appendix F.1. Minimum and Maximum Values for Emitter Parameters

Each emitter is identified by the following parameters which are stored in a parameter array, one for each emitter. See Figure 1 in section B.1 for the order in which these parameters are stored. This table includes the maximum and minimum values for each parameter.

EMITTER PARAMETER	MINIMUM	MAXIMUM
X Displacement of Emitter and Signal Sorter	2000 m	22000 m
Y Displacement of Emitter and Signal Sorter	-22000 m	22000 m
Z Displacement of Emitter and Signal Sorter	0 m	1000 m
Emitter Power (includes antenna gain)	30 db	80 db
Mainlobe Size	.09	.32
Sidelobe Loss	15 db	30 db
Initial Antenna Angle	0	---
Antenna Angle	-1.5 radians	1.5 radians
Scan Rate	.5 Hz	2 Hz
PRI	500 microscs	7900 microscs
Pulse Width	1 microscs	4 microscs
Frequency	$2*10^9$ Hz	$12*10^9$ Hz
On Time	0 secs	max. sep. time
Off Time	---	Runtime
TOA of pulse at Antenna	On Time	---
DOA of pulse at Antenna	360	---
X Velocity of Signal Sorter	-300 m/sec	300 m/sec
Y Velocity of Signal Sorter	-300 m/sec	300 m/sec
Z Velocity of Signal Sorter	0	---

--- indicates that the value is not applicable to the parameter.

F.2. Statistical Package

For the purpose of comparing simulation runs and to determine the state of the system for particular runs, pertinent data is output by the STATS subroutine at specified intervals. Subroutines with the Q-prefx are used to output data concerning queue lengths of the various buffers and tables analyzing averages for various parameters of the system. The information from these routines can be used to determine the system workload for any particular environment and run-time.

Queue analyses include histograms for maximum queue lenght during the time interval data is collected, minimum and maximum values of the queue and the length of time, maximum and minimum, that these values were true. For all tables generated, the sum, average, maximum, minimum and total frequency of the arguments are calculated. For a sample of both a table and a histogram/table from these routines, see tables 2.1 and 2.2.

EXPLANATION OF OUTPUT VARIABLES - QTABLES:

TIMING GATE	- period during which queue statistics are collected
FIRST ARRIVAL	- time of arrival of first emitter on which data is collected
LAST ARRIVAL	- time of arrival of last emitter on which data is collected
SUM OF ARG'S	- total of all data collected for a particular parameter, e.g. PRIs
AV'G OF ARG'S	- SUM OF ARG'S/TOTAL FREQUENCY
TOTAL FREQ	- number of times a parameter is observed
MINIMUM ARG	- smallest parameter observed
MAXIMUM ARG	- largest parameter observed
UPPER LIMIT	- for other than the upper boundary, this value is the range less than the previous value and less than or equal to the current value; the upper boundary also includes all values that is greater than the current value
OBSV FREQ	- the number of times the parameter was in the range of the UPPER LIMIT
% OF TOTAL	- percentage of TOTAL FREQ that the parameter was in the range of the current UPPER LIMIT
CUM. %	- percentage of TOTAL FREQ that parameters are in range of the current UPPER LIMIT and less
DELTA-T	- minimum and maximum amount of time that the queue remained a constant size
DELTA-T: MINIMUM	- the shortest time interval for the queue length specified by Q-LEN
DELTA-T: MAXIMUM	- the longest time interval for the queue length specified by Q-LEN
DELTA-T: TIME	- time at which the queue remained at size Q-LEN for the specified DELTA-T
Q-LENGTH	- minimum and maximum size of the queue
Q-LEN: DELTA-T	- the period of time during which the queue remained at the specified size
Q-LEN: TIME	- time at which the queue reached the specified size

See table 2.3 for a sample listing from the STATS subroutine. Output includes the maximum number of emitters in each bine of main memory, in addition to the current size; maximum number of emitters that were in any bin of the LIST at one time, and current LIST sizes; and the total and average contents of the LIST for each printing. The total number of emitters in the main memory file should be equal to the number of emitters in the environment, unless an emitter changes DOA or frequency outside the expected range. In that case, the emitter would appear as a new emitter, and would be entered in the file as such.

EXPLANATION OF OUTPUT VARIABLES - STATS:

NMCNT	- number of nomatches in the CAM not due to DOA or frequency drift: strict nomatches
MCNT	- number of emitters matched in the CAM
MAX	- maximum delay through the system of a data word
CAMB	- maximum size of the input buffer to the associative-processor (output from receiver for CAM match/nomatch)
MPPB	- maximum size of the input buffer to the microprocessor-array input buffer
MTIME	- time at which MAXimum delay occurred
NCWL	- number of words loaded into the CAM
MLCAMBUF	- maximum size of the output buffer from the M-A
IDOAD	- nomatches due to DOA drifts within +-1
ICFD	- nomatches due to frequency drifts within +-1
NEA	- number of external arrivals: data passed to the CAM from the receiver system
NIA	- number of internal arrivals: data passed to the M-A due to nomatch in the CAM, includes new emitters, DOA and frequency drifts (all nomatches)
INLT2	- number of updates in the file for drifting parameters
INLT3	- number of updates in the file with no drift

Table F.2.1. Statistical Package Output

TABLE 4 AVERAGE PRI OF 100 EMITTERS

TIMING GATE ON	TIMING GATE OFF	FIRST ARRIVAL	LAST ARRIVAL
0.00000E+00	0.11000E+00	0.10000E-04	0.10000E-02

SUM OF ARGUMENTS	AVERAGE OF ARGUMENTS	TOTAL FREQ	MINIMUN ARGUMENT	MAXIMUM ARGUMENT
0.42279E+00	0.42279E-02	100	0.50000E-03	0.79000E-02

UPPER LIMIT	OBSV. FREQ	% OF TOTAL	CUM. %
0.00000	0	0.00	0.00
0.00020	0	0.00	0.00
0.00040	0	0.00	0.00
0.00060	1	1.00	1.00
0.00080	4	4.00	5.00
0.00100	3	3.00	8.00
0.00120	1	1.00	9.00
0.00140	5	5.00	14.00
0.00160	2	2.00	16.00
0.00180	2	2.00	18.00
0.00200	0	0.00	18.00
0.00220	2	2.00	20.00
0.00240	3	3.00	23.00
0.00260	4	4.00	27.00
0.00280	2	2.00	29.00
0.00300	2	2.00	31.00
0.00320	2	2.00	33.00
0.00340	3	3.00	36.00
0.00360	2	2.00	38.00
0.00380	5	5.00	43.00
0.00400	2	2.00	45.00
0.00420	0	0.00	45.00
0.00440	3	3.00	48.00
0.00460	3	3.00	51.00
0.00480	3	3.00	54.00
0.00500	4	4.00	58.00
0.00520	5	5.00	63.00
0.00540	5	5.00	68.00
0.00560	2	2.00	70.00
0.00580	7	7.00	77.00
0.00600	1	1.00	78.00
0.00620	3	3.00	81.00
0.00640	2	2.00	83.00
0.00660	3	3.00	86.00
0.00680	2	2.00	88.00
0.00700	3	3.00	91.00
0.00720	0	0.00	91.00
0.00740	3	3.00	94.00
0.00760	3	3.00	97.00
0.00780	0	0.00	97.00
0.00800	3	3.00	100.00

REMAINING FREQUENCIES ALL ZERO

Table F.2.2. Statistical Package Output

TABLE 2 MICROPROCESSORS BUFFER (LMISIZ)

TIMING GATE ON	TIMING GATE OFF	FIRST ARRIVAL	LAST ARRIVAL	TOTAL TIME
0.00000E+00	0.11000E+01	0.00000E+00	0.99992E+00	0.99992E+00

DELTA-T MINIMUM	DELTA-T Q-LEN	DELTA-T TIME	DELTA-T MAXIMUM	DELTA-T Q-LEN	DELTA-T TIME
0.00000E+00	0	0.00000E+00	0.58047E-01	1	0.21919E+00

Q-LENGTH MIN	Q-LENGTH DELTA-T	Q-LENGTH TIME	Q-LENGTH MAX	Q-LENGTH DELTA-T	Q-LENGTH TIME
0	0.00000E+00	0.00000E+00	4	0.52201E-04	0.19813E-01

SUM OF LENGTH*TIME	AVERAGE OF LENGTH*TIME	TOTAL FREQ
0.10021E+01	0.10022E+01	2022

UPPER LIMIT	OBSV. FREQ	% OF TOTAL	CUM. %
0	947	46.83	46.83
1	1000	49.46	96.29
2	62	3.07	99.36
3	11	0.54	99.90
4	2	0.10	100.00

REMAINING FREQUENCIES ALL ZERO

MAXIMUM Q-LENGTH DURING TIME INTERVAL

0.00000E+00		0
0.50000E-01	**********	4
0.10000E+00	**********	2
0.15000E+00	******	1
0.20000E+00	******	1
0.25000E+00	******	1
0.30000E+00	******	1
0.35000E+00	******	1
0.40000E+00	******	1
0.45000E+00	******	1
0.50000E+00	******	1
0.55000E+00	******	1
0.60000E+00	******	1
0.65000E+00	******	1
0.70000E+00	******	1
0.75000E+00	**********	2
0.80000E+00	******	1
0.85000E+00	**********	2
0.90000E+00	**********	2
0.95000E+00	**********	2
0.10000E+01	******	1

REMAINING FREQUENCIES ALL ZERO

Table F.2.3. Statistical Package Output

```
APRI= 0.42279E-02


TIME =  1.00 SEC.   (PRINT INCR=  1.000)

NMCNT=         6  MCNT=     36482  MAX=     364  CAMB=  3  MPPB=  4
 MTIME= 0.020068  NCWL=     36539  MLCAMBUF=  3  IDOAD=     49
 ICFD=     0  NEA=     36853  NIA=     370  INLT1=    0  INLT2=  51  INLT3=  104
   0  2  3  1  2  3  2  2  1  3  3  5  3  5  2  1
   3  3  2  2  2  4  2  2  3  1  0  2  2  1  1  2
   3  2  1  5  2  1  1  3  4  2  1  0  0  0  3  4
   4  3  2  2  4  1  0  4  2  0  1  0  0  0  0  0
MAIN MEMORY SIZES:
   0  0  3  1  2  2  1  2  0  3  2  3  3  4  2  0
   3  2  2  1  2  4  1  2  2  1  0  2  2  1  1  1
   3  2  0  4  2  0  1  2  4  1  1  0  0  0  1  3
   4  3  1  2  4  1  0  3  2  0  1  0  0  0  0  0
    TOTAL EMITERS IN MPPR=100
  17 16 17 16 17 17 17 18 17 18 16 17 20 15 16 17
  15 16 17 20 17 21 16 17 18 16 17 18 16 18 17 18
LIST SIZES:
   0  0  7  8  6  7  6  3  5  6  7  5  2  3  4  4
   5  3  2  2  3  3  1  1  0  3  0  0  0  1  0  0
TOTAL CONTENTS=  97    AVERAGE CONTENTS=  3.03
```

APPENDIX F.3. Emitter Environment - 300 Emitters

The following table contains the DOA, frequency and PRI values for an environment of 300 emitters. The list is sorted by frequency. The environment contains no exotic emitters. These are the values for all 300 emitters immediately after all emitters have been turned on.

mod # - module in main memory which corresponds to the DOA of an emitter.
item# - order of the item in the main memory module. For instance, if mod # is 14 and item# is 6, the emitter was the sixth emitter to be entered into that module.
freq - frequency of the emitter in Hz
pri - pulse repetition interval of the emitter in microseconds

mod #	35	item#	2	freq	5	pri	6025
mod #	55	item#	4	freq	25	pri	5476
mod #	61	item#	1	freq	27	pri	4451
mod #	20	item#	7	freq	72	pri	7623
mod #	18	item#	4	freq	72	pri	2931
mod #	58	item#	3	freq	78	pri	4317
mod #	41	item#	5	freq	90	pri	5546
mod #	9	item#	5	freq	147	pri	1065
mod #	21	item#	3	freq	158	pri	7900
mod #	5	item#	1	freq	166	pri	5267
mod #	38	item#	2	freq	168	pri	4906
mod #	14	item#	6	freq	170	pri	6946
mod #	4	item#	4	freq	182	pri	3406
mod #	56	item#	4	freq	207	pri	2434
mod #	46	item#	6	freq	209	pri	4393
mod #	14	item#	4	freq	228	pri	7204
mod #	36	item#	5	freq	241	pri	5405
mod #	31	item#	2	freq	247	pri	5079
mod #	37	item#	1	freq	252	pri	2975
mod #	8	item#	1	freq	279	pri	6103
mod #	3	item#	1	freq	281	pri	7096
mod #	53	item#	4	freq	315	pri	7490
mod #	15	item#	2	freq	319	pri	1755
mod #	50	item#	2	freq	325	pri	5149
mod #	21	item#	4	freq	363	pri	1824
mod #	34	item#	2	freq	408	pri	5398
mod #	20	item#	2	freq	409	pri	7623
mod #	38	item#	3	freq	410	pri	2711
mod #	10	item#	12	freq	428	pri	1614
mod #	51	item#	2	freq	433	pri	5360
mod #	30	item#	3	freq	453	pri	5273
mod #	18	item#	11	freq	453	pri	6643
mod #	29	item#	1	freq	476	pri	3079
mod #	33	item#	6	freq	513	pri	559
mod #	17	item#	1	freq	516	pri	6300
mod #	58	item#	1	freq	533	pri	2321
mod #	10	item#	11	freq	536	pri	1126
mod #	55	item#	6	freq	544	pri	6667
mod #	50	item#	4	freq	546	pri	2953

mod #	55	item#	1	freq	568	pri	737
mod #	46	item#	3	freq	571	pri	3786
mod #	49	item#	8	freq	572	pri	4472
mod #	10	item#	7	freq	602	pri	7393
mod #	24	item#	4	freq	606	pri	6152
mod #	3	item#	5	freq	632	pri	3286
mod #	19	item#	7	freq	634	pri	4459
mod #	39	item#	1	freq	649	pri	2575
mod #	37	item#	3	freq	650	pri	1412
mod #	32	item#	5	freq	652	pri	3640
mod #	7	item#	1	freq	659	pri	3908
mod #	49	item#	2	freq	671	pri	3498
mod #	23	item#	4	freq	678	pri	2999
mod #	16	item#	3	freq	699	pri	5217
mod #	35	item#	3	freq	705	pri	2205
mod #	33	item#	2	freq	732	pri	2394
mod #	31	item#	1	freq	736	pri	4070
mod #	6	item#	3	freq	769	pri	1514
mod #	17	item#	4	freq	777	pri	3588
mod #	18	item#	2	freq	787	pri	2111
mod #	14	item#	10	freq	795	pri	1575
mod #	45	item#	5	freq	808	pri	5096
mod #	15	item#	1	freq	894	pri	1811
mod #	18	item#	7	freq	900	pri	4957
mod #	32	item#	3	freq	908	pri	1802
mod #	60	item#	1	freq	925	pri	6949
mod #	16	item#	5	freq	928	pri	2740
mod #	16	item#	7	freq	947	pri	6578
mod #	18	item#	5	freq	953	pri	1151
mod #	56	item#	2	freq	953	pri	715
mod #	9	item#	2	freq	975	pri	3514
mod #	17	item#	5	freq	984	pri	1689
mod #	16	item#	9	freq	987	pri	5170
mod #	49	item#	5	freq	991	pri	2123
mod #	19	item#	4	freq	994	pri	7675
mod #	48	item#	1	freq	1003	pri	5952
mod #	25	item#	4	freq	1019	pri	3050
mod #	6	item#	2	freq	1037	pri	5123
mod #	14	item#	2	freq	1045	pri	4839
mod #	26	item#	2	freq	1051	pri	5463
mod #	43	item#	4	freq	1066	pri	6904
mod #	46	item#	5	freq	1099	pri	616
mod #	10	item#	1	freq	1122	pri	3357
mod #	24	item#	1	freq	1147	pri	3470
mod #	11	item#	3	freq	1169	pri	5533
mod #	14	item#	5	freq	1177	pri	4324
mod #	54	item#	3	freq	1180	pri	5459
mod #	6	item#	1	freq	1188	pri	2558
mod #	55	item#	5	freq	1198	pri	4915
mod #	10	item#	8	freq	1218	pri	5749
mod #	25	item#	2	freq	1224	pri	6927
mod #	33	item#	1	freq	1232	pri	4941
mod #	47	item#	1	freq	1238	pri	3922
mod #	36	item#	7	freq	1248	pri	7525
mod #	27	item#	2	freq	1255	pri	2927
mod #	52	item#	1	freq	1263	pri	7612
mod #	57	item#	1	freq	1270	pri	1741
mod #	3	item#	3	freq	1270	pri	2473

mod #	23	item#	2	freq	1284	pri	2931
mod #	25	item#	3	freq	1287	pri	1170
mod #	27	item#	3	freq	1324	pri	1105
mod #	22	item#	5	freq	1329	pri	5405
mod #	28	item#	2	freq	1333	pri	4070
mod #	17	item#	3	freq	1355	pri	4403
mod #	36	item#	4	freq	1363	pri	3765
mod #	25	item#	1	freq	1369	pri	3926
mod #	49	item#	7	freq	1373	pri	2031
mod #	10	item#	4	freq	1379	pri	1493
mod #	31	item#	4	freq	1389	pri	2486
mod #	10	item#	9	freq	1411	pri	2492
mod #	52	item#	2	freq	1413	pri	1071
mod #	33	item#	3	freq	1415	pri	1282
mod #	16	item#	1	freq	1416	pri	7516
mod #	52	item#	6	freq	1422	pri	5715
mod #	29	item#	4	freq	1434	pri	4435
mod #	49	item#	9	freq	1435	pri	5511
mod #	13	item#	3	freq	1448	pri	6652
mod #	33	item#	4	freq	1469	pri	5139
mod #	29	item#	3	freq	1486	pri	967
mod #	52	item#	3	freq	1500	pri	1344
mod #	45	item#	7	freq	1510	pri	3222
mod #	50	item#	5	freq	1554	pri	1905
mod #	32	item#	1	freq	1594	pri	2842
mod #	45	item#	6	freq	1605	pri	5753
mod #	31	item#	5	freq	1606	pri	3277
mod #	44	item#	4	freq	1610	pri	5483
mod #	17	item#	2	freq	1612	pri	1292
mod #	53	item#	3	freq	1624	pri	6009
mod #	57	item#	6	freq	1627	pri	4322
mod #	13	item#	5	freq	1636	pri	5574
mod #	44	item#	2	freq	1637	pri	3393
mod #	48	item#	3	freq	1648	pri	5269
mod #	11	item#	1	freq	1652	pri	1269
mod #	43	item#	1	freq	1669	pri	7900
mod #	11	item#	2	freq	1682	pri	6975
mod #	54	item#	2	freq	1685	pri	5105
mod #	43	item#	6	freq	1697	pri	7670
mod #	9	item#	7	freq	1703	pri	2867
mod #	25	item#	6	freq	1705	pri	7094
mod #	37	item#	4	freq	1712	pri	7634
mod #	18	item#	8	freq	1713	pri	5169
mod #	23	item#	3	freq	1718	pri	2089
mod #	36	item#	6	freq	1759	pri	2934
mod #	16	item#	8	freq	1759	pri	4035
mod #	18	item#	9	freq	1759	pri	7644
mod #	9	item#	8	freq	1784	pri	4316
mod #	21	item#	1	freq	1789	pri	3213
mod #	44	item#	5	freq	1798	pri	6629
mod #	57	item#	2	freq	1805	pri	6405
mod #	13	item#	4	freq	1809	pri	5825
mod #	44	item#	1	freq	1822	pri	2612
mod #	27	item#	1	freq	1839	pri	7171
mod #	24	item#	2	freq	1849	pri	4955
mod #	51	item#	5	freq	1862	pri	2734
mod #	45	item#	4	freq	1912	pri	3717
mod #	3	item#	4	freq	1923	pri	4754

mod #	2	item#	1	freq	1958	pri	1309
mod #	21	item#	5	freq	1962	pri	5185
mod #	4	item#	2	freq	1991	pri	6780
mod #	19	item#	5	freq	1991	pri	7323
mod #	53	item#	2	freq	1994	pri	1929
mod #	28	item#	3	freq	2002	pri	6535
mod #	43	item#	2	freq	2033	pri	6092
mod #	54	item#	1	freq	2041	pri	1721
mod #	30	item#	2	freq	2056	pri	2850
mod #	52	item#	7	freq	2066	pri	823
mod #	26	item#	1	freq	2069	pri	2615
mod #	36	item#	8	freq	2076	pri	4970
mod #	57	item#	4	freq	2081	pri	5543
mod #	30	item#	5	freq	2087	pri	6633
mod #	50	item#	3	freq	2135	pri	6082
mod #	8	item#	4	freq	2138	pri	6555
mod #	16	item#	2	freq	2145	pri	3431
mod #	56	item#	3	freq	2157	pri	3659
mod #	19	item#	2	freq	2167	pri	7474
mod #	18	item#	6	freq	2179	pri	7468
mod #	34	item#	1	freq	2181	pri	6269
mod #	23	item#	5	freq	2182	pri	3704
mod #	8	item#	6	freq	2202	pri	983
mod #	23	item#	6	freq	2210	pri	3297
mod #	25	item#	5	freq	2221	pri	1222
mod #	50	item#	1	freq	2251	pri	6411
mod #	16	item#	6	freq	2258	pri	2911
mod #	22	item#	2	freq	2270	pri	5738
mod #	25	item#	7	freq	2284	pri	6622
mod #	58	item#	2	freq	2311	pri	2240
mod #	12	item#	2	freq	2316	pri	3768
mod #	41	item#	4	freq	2329	pri	7197
mod #	24	item#	3	freq	2365	pri	5579
mod #	9	item#	3	freq	2377	pri	5330
mod #	16	item#	4	freq	2380	pri	7900
mod #	12	item#	4	freq	2393	pri	2905
mod #	38	item#	1	freq	2395	pri	3490
mod #	55	item#	2	freq	2419	pri	6666
mod #	19	item#	1	freq	2429	pri	5788
mod #	2	item#	2	freq	2451	pri	2912
mod #	57	item#	5	freq	2467	pri	4574
mod #	36	item#	1	freq	2482	pri	3280
mod #	47	item#	4	freq	2487	pri	3770
mod #	9	item#	1	freq	2502	pri	4986
mod #	8	item#	5	freq	2518	pri	1740
mod #	18	item#	3	freq	2526	pri	5602
mod #	3	item#	2	freq	2576	pri	5082
mod #	20	item#	1	freq	2582	pri	2583
mod #	41	item#	1	freq	2597	pri	4585
mod #	41	item#	3	freq	2613	pri	3633
mod #	19	item#	6	freq	2672	pri	4415
mod #	46	item#	7	freq	2687	pri	5423
mod #	10	item#	10	freq	2690	pri	6828
mod #	3	item#	6	freq	2705	pri	3539
mod #	13	item#	2	freq	2724	pri	6201
mod #	10	item#	3	freq	2747	pri	618
mod #	47	item#	7	freq	2753	pri	5034
mod #	52	item#	5	freq	2766	pri	1342

mod	#	12	item#	1	freq	2770	pri	3736
mod	#	46	item#	2	freq	2795	pri	5692
mod	#	35	item#	1	freq	2800	pri	7450
mod	#	47	item#	3	freq	2806	pri	7007
mod	#	4	item#	1	freq	2824	pri	7900
mod	#	32	item#	2	freq	2842	pri	6370
mod	#	30	item#	1	freq	2843	pri	500
mod	#	47	item#	5	freq	2844	pri	1489
mod	#	43	item#	3	freq	2845	pri	6832
mod	#	57	item#	7	freq	2847	pri	3608
mod	#	22	item#	6	freq	2857	pri	6704
mod	#	20	item#	5	freq	2906	pri	6206
mod	#	40	item#	4	freq	2953	pri	5186
mod	#	42	item#	1	freq	2957	pri	3429
mod	#	20	item#	3	freq	2979	pri	3181
mod	#	14	item#	9	freq	2982	pri	4781
mod	#	47	item#	6	freq	2989	pri	1241
mod	#	51	item#	1	freq	3001	pri	5047
mod	#	14	item#	7	freq	3021	pri	3360
mod	#	42	item#	2	freq	3029	pri	7064
mod	#	20	item#	4	freq	3038	pri	6217
mod	#	34	item#	3	freq	3055	pri	5936
mod	#	8	item#	2	freq	3062	pri	500
mod	#	46	item#	1	freq	3074	pri	4111
mod	#	12	item#	3	freq	3189	pri	6711
mod	#	28	item#	1	freq	3204	pri	7462
mod	#	8	item#	3	freq	3220	pri	2775
mod	#	20	item#	6	freq	3221	pri	1634
mod	#	29	item#	2	freq	3227	pri	7900
mod	#	18	item#	10	freq	3247	pri	5963
mod	#	18	item#	1	freq	3250	pri	3044
mod	#	49	item#	1	freq	3254	pri	2979
mod	#	31	item#	3	freq	3262	pri	3154
mod	#	56	item#	1	freq	3279	pri	7189
mod	#	21	item#	2	freq	3280	pri	4013
mod	#	4	item#	3	freq	3298	pri	1820
mod	#	48	item#	2	freq	3326	pri	1473
mod	#	52	item#	8	freq	3335	pri	2024
mod	#	22	item#	1	freq	3345	pri	3343
mod	#	38	item#	4	freq	3361	pri	3495
mod	#	41	item#	2	freq	3375	pri	904
mod	#	10	item#	6	freq	3383	pri	2862
mod	#	27	item#	4	freq	3402	pri	4773
mod	#	57	item#	3	freq	3457	pri	4070
mod	#	49	item#	3	freq	3486	pri	766
mod	#	45	item#	3	freq	3491	pri	5986
mod	#	51	item#	3	freq	3493	pri	3641
mod	#	45	item#	1	freq	3567	pri	6063
mod	#	11	item#	4	freq	3604	pri	3337
mod	#	10	item#	5	freq	3614	pri	5007
mod	#	55	item#	3	freq	3651	pri	6300
mod	#	53	item#	1	freq	3652	pri	1372
mod	#	53	item#	5	freq	3655	pri	677
mod	#	9	item#	4	freq	3667	pri	4565
mod	#	6	item#	4	freq	3667	pri	3036
mod	#	38	item#	3	freq	3692	pri	2250

mod #	43	item#	5	freq	3692	pri	3922
mod #	10	item#	2	freq	3696	pri	948
mod #	37	item#	2	freq	3699	pri	803
mod #	14	item#	3	freq	3700	pri	2627
mod #	25	item#	8	freq	3707	pri	5289
mod #	36	item#	2	freq	3778	pri	2990
mod #	49	item#	6	freq	3790	pri	4721
mod #	14	item#	1	freq	3796	pri	6243
mod #	7	item#	2	freq	3829	pri	2744
mod #	37	item#	5	freq	3840	pri	4157
mod #	39	item#	2	freq	3853	pri	6540
mod #	32	item#	4	freq	3858	pri	4922
mod #	44	item#	3	freq	3928	pri	6664
mod #	47	item#	2	freq	3947	pri	4948
mod #	33	item#	5	freq	3967	pri	6859
mod #	49	item#	4	freq	3977	pri	3697
mod #	19	item#	3	freq	3986	pri	5564
mod #	52	item#	4	freq	3991	pri	4473
mod #	14	item#	8	freq	3999	pri	6290
mod #	30	item#	4	freq	4030	pri	500
mod #	9	item#	6	freq	4032	pri	5498
mod #	22	item#	3	freq	4045	pri	6765
mod #	22	item#	4	freq	4054	pri	6323
mod #	55	item#	7	freq	4054	pri	7900
mod #	23	item#	1	freq	4057	pri	2008
mod #	59	item#	1	freq	4086	pri	4937
mod #	51	item#	4	freq	4095	pri	3293

APPENDIX F.4. Environments for 100 and 102 Emitters

This appendix contains the values of DOA, frequency and DOA for two environments, an environment of 100 emitters and an environment of 102 emitters. Sections F.4.a. and F.4.b. contain the values for the 100 emitter environment. Section F.4.a. is sorted by frequency while section F.4.b. is the same data sorted by PRI. Sections F.4.c. and F.4.d. contain the values for an environment of 102 emitters by frequency and PRI, respectively. The only difference in the input values that produced the two environments was the number of emitters.

F.4.a. 100 Emitters by Frequency

mod #	48	item#	3	freq	5	pri	7565
mod #	33	item#	3	freq	8	pri	4271
mod #	18	item#	2	freq	66	pri	922
mod #	36	item#	1	freq	132	pri	3280
mod #	6	item#	2	freq	148	pri	7334
mod #	53	item#	4	freq	168	pri	2860
mod #	29	item#	2	freq	248	pri	1248
mod #	21	item#	2	freq	259	pri	2580
mod #	17	item#	3	freq	317	pri	2788
mod #	51	item#	1	freq	371	pri	2538
mod #	36	item#	2	freq	483	pri	3170
mod #	41	item#	3	freq	490	pri	7900
mod #	15	item#	1	freq	550	pri	5897
mod #	40	item#	1	freq	688	pri	5773
mod #	41	item#	4	freq	726	pri	2690
mod #	52	item#	2	freq	791	pri	7900
mod #	17	item#	1	freq	835	pri	6500
mod #	37	item#	1	freq	870	pri	7900
mod #	36	item#	3	freq	880	pri	6297
mod #	22	item#	4	freq	892	pri	2119
mod #	28	item#	2	freq	1008	pri	1441
mod #	14	item#	4	freq	1139	pri	4767
mod #	33	item#	2	freq	1153	pri	5108
mod #	24	item#	2	freq	1169	pri	4948
mod #	20	item#	1	freq	1171	pri	2294
mod #	34	item#	2	freq	1178	pri	7376
mod #	41	item#	2	freq	1259	pri	5227
mod #	22	item#	1	freq	1296	pri	3634
mod #	40	item#	2	freq	1301	pri	5179
mod #	50	item#	1	freq	1308	pri	4421
mod #	22	item#	3	freq	1346	pri	920
mod #	52	item#	1	freq	1370	pri	5795
mod #	25	item#	2	freq	1372	pri	3537
mod #	10	item#	1	freq	1378	pri	1700
mod #	5	item#	1	freq	1408	pri	5652
mod #	4	item#	1	freq	1412	pri	4424
mod #	14	item#	2	freq	1427	pri	5263
mod #	15	item#	2	freq	1469	pri	4214
mod #	26	item#	1	freq	1484	pri	4866
mod #	53	item#	3	freq	1502	pri	4717
mod #	3	item#	1	freq	1539	pri	6087
mod #	5	item#	2	freq	1574	pri	2519
mod #	17	item#	2	freq	1581	pri	2206
mod #	34	item#	1	freq	1639	pri	3738

mod #	33	item#	1	freq	1655	pri	7232
mod #	31	item#	1	freq	1660	pri	4806
mod #	49	item#	2	freq	1668	pri	5716
mod #	54	item#	1	freq	1791	pri	6091
mod #	49	item#	4	freq	1887	pri	5353
mod #	30	item#	1	freq	1894	pri	5435
mod #	59	item#	1	freq	1897	pri	5617
mod #	22	item#	2	freq	1944	pri	659
mod #	11	item#	1	freq	1976	pri	2830
mod #	7	item#	1	freq	1976	pri	1216
mod #	49	item#	3	freq	2000	pri	6898
mod #	24	item#	1	freq	2178	pri	6001
mod #	3	item#	3	freq	2261	pri	6558
mod #	49	item#	1	freq	2402	pri	7471
mod #	50	item#	2	freq	2462	pri	943
mod #	39	item#	1	freq	2483	pri	4912
mod #	3	item#	2	freq	2495	pri	3248
mod #	13	item#	1	freq	2510	pri	2542
mod #	13	item#	3	freq	2534	pri	1238
mod #	57	item#	1	freq	2536	pri	5663
mod #	53	item#	2	freq	2559	pri	1321
mod #	25	item#	1	freq	2639	pri	642
mod #	8	item#	2	freq	2754	pri	1422
mod #	56	item#	2	freq	2803	pri	5282
mod #	19	item#	1	freq	2824	pri	6876
mod #	56	item#	1	freq	2899	pri	3806
mod #	50	item#	3	freq	2926	pri	3357
mod #	8	item#	1	freq	3017	pri	500
mod #	43	item#	1	freq	3079	pri	3667
mod #	47	item#	1	freq	3095	pri	4529
mod #	10	item#	3	freq	3101	pri	6763
mod #	12	item#	2	freq	3155	pri	2071
mod #	23	item#	1	freq	3164	pri	4717
mod #	14	item#	1	freq	3179	pri	6990
mod #	48	item#	1	freq	3193	pri	1642
mod #	21	item#	1	freq	3210	pri	5158
mod #	36	item#	4	freq	3211	pri	3078
mod #	42	item#	1	freq	3268	pri	4239
mod #	32	item#	1	freq	3351	pri	5086
mod #	12	item#	3	freq	3405	pri	3936
mod #	19	item#	2	freq	3412	pri	7571
mod #	6	item#	1	freq	3476	pri	3587
mod #	37	item#	2	freq	3557	pri	6652
mod #	14	item#	3	freq	3651	pri	6527
mod #	28	item#	1	freq	3692	pri	3677
mod #	12	item#	1	freq	3702	pri	2370
mod #	18	item#	1	freq	3847	pri	1160
mod #	41	item#	1	freq	3860	pri	5016
mod #	56	item#	3	freq	3883	pri	5591
mod #	10	item#	2	freq	3936	pri	6254
mod #	53	item#	1	freq	3942	pri	5271
mod #	13	item#	2	freq	3989	pri	3674
mod #	29	item#	1	freq	4026	pri	618
mod #	57	item#	2	freq	4089	pri	5733

F.4.b. 100 Emitters by PRI

mod #	8	item#	1	freq	3017	pri	500
mod #	29	item#	1	freq	4026	pri	618
mod #	25	item#	1	freq	2639	pri	642
mod #	22	item#	2	freq	1944	pri	659
mod #	48	item#	2	freq	3037	pri	728
mod #	22	item#	3	freq	1346	pri	920
mod #	18	item#	2	freq	66	pri	922
mod #	50	item#	2	freq	2462	pri	943
mod #	18	item#	1	freq	3847	pri	1160
mod #	7	item#	1	freq	1976	pri	1216
mod #	13	item#	3	freq	2534	pri	1238
mod #	29	item#	2	freq	248	pri	1248
mod #	11	item#	2	freq	3058	pri	1270
mod #	53	item#	2	freq	2559	pri	1321
mod #	8	item#	2	freq	2754	pri	1422
mod #	28	item#	2	freq	1008	pri	1441
mod #	48	item#	1	freq	3193	pri	1642
mod #	10	item#	1	freq	1378	pri	1700
mod #	12	item#	2	freq	3155	pri	2071
mod #	22	item#	4	freq	892	pri	2119
mod #	17	item#	2	freq	1581	pri	2206
mod #	20	item#	1	freq	1171	pri	2294
mod #	12	item#	1	freq	3702	pri	2370
mod #	5	item#	2	freq	1574	pri	2519
mod #	51	item#	1	freq	371	pri	2538
mod #	13	item#	1	freq	2510	pri	2542
mod #	21	item#	2	freq	259	pri	2580
mod #	41	item#	4	freq	726	pri	2690
mod #	17	item#	3	freq	317	pri	2788
mod #	11	item#	1	freq	1976	pri	2830
mod #	53	item#	4	freq	168	pri	2860
mod #	36	item#	4	freq	3211	pri	3078
mod #	36	item#	2	freq	483	pri	3170
mod #	3	item#	2	freq	2495	pri	3248
mod #	36	item#	1	freq	132	pri	3280
mod #	50	item#	3	freq	2926	pri	3357
mod #	25	item#	2	freq	1372	pri	3537
mod #	6	item#	1	freq	3476	pri	3587
mod #	22	item#	1	freq	1296	pri	3634
mod #	43	item#	1	freq	3079	pri	3667
mod #	13	item#	2	freq	3989	pri	3674
mod #	28	item#	1	freq	3692	pri	3677
mod #	34	item#	1	freq	1639	pri	3738
mod #	56	item#	1	freq	2899	pri	3806
mod #	12	item#	3	freq	3405	pri	3936
mod #	15	item#	2	freq	1469	pri	4214
mod #	42	item#	1	freq	3268	pri	4239
mod #	33	item#	3	freq	8	pri	4271
mod #	50	item#	1	freq	1303	pri	4421
mod #	4	item#	1	freq	1412	pri	4424
mod #	47	item#	1	freq	3095	pri	4529
mod #	53	item#	3	freq	1502	pri	4717
mod #	23	item#	1	freq	3164	pri	4717
mod #	14	item#	4	freq	1139	pri	4767
mod #	31	item#	1	freq	1660	pri	4806
mod #	26	item#	1	freq	1484	pri	4866

```
mod # 39 item#  1 freq  2483  pri  4912
mod # 24 item#  2 freq  1169  pri  4948
mod # 41 item#  1 freq  3860  pri  5016
mod # 32 item#  1 freq  3351  pri  5086
mod # 33 item#  2 freq  1153  pri  5108
mod # 21 item#  1 freq  3210  pri  5158
mod # 40 item#  2 freq  1301  pri  5179
mod # 41 item#  2 freq  1259  pri  5227
mod # 14 item#  2 freq  1427  pri  5263
mod # 53 item#  1 freq  3942  pri  5271
mod # 56 item#  2 freq  2803  pri  5282
mod # 49 item#  4 freq  1887  pri  5353
mod # 30 item#  1 freq  1894  pri  5435
mod # 56 item#  3 freq  3883  pri  5591
mod # 59 item#  1 freq  1897  pri  5617
mod #  5 item#  1 freq  1408  pri  5652
mod # 57 item#  1 freq  2536  pri  5663
mod # 49 item#  2 freq  1668  pri  5716
mod # 57 item#  2 freq  4089  pri  5733
mod # 40 item#  1 freq   688  pri  5773
mod # 52 item#  1 freq  1370  pri  5795
mod # 15 item#  1 freq   550  pri  5897
mod # 24 item#  1 freq  2178  pri  6001
mod #  3 item#  1 freq  1539  pri  6087
mod # 54 item#  1 freq  1791  pri  6091
mod # 10 item#  2 freq  3936  pri  6254
mod # 36 item#  3 freq   880  pri  6297
mod # 17 item#  1 freq   835  pri  6500
mod # 14 item#  3 freq  3651  pri  6527
mod #  3 item#  3 freq  2261  pri  6558
mod # 37 item#  2 freq  3557  pri  6652
mod # 10 item#  3 freq  3101  pri  6763
mod # 19 item#  1 freq  2824  pri  6876
mod # 49 item#  3 freq  2000  pri  6898
mod # 14 item#  1 freq  3179  pri  6990
mod # 33 item#  1 freq  1655  pri  7232
mod #  6 item#  2 freq   148  pri  7334
mod # 34 item#  2 freq  1178  pri  7376
mod # 49 item#  1 freq  2402  pri  7471
mod # 48 item#  3 freq     5  pri  7565
mod # 19 item#  2 freq  3412  pri  7571
mod # 52 item#  2 freq   791  pri  7900
mod # 41 item#  3 freq   490  pri  7900
mod # 37 item#  1 freq   870  pri  7900
```

F.4.c. 102 Emitters by Frequency

```
mod # 35 item#  2 freq     6  pri  6025
mod #  1 item#  2 freq   248  pri  5079
mod # 37 item#  1 freq   253  pri  2975
mod #  7 item#  3 freq   280  pri  6103
mod #  3 item#  3 freq   282  pri  7096
mod # 20 item#  2 freq   410  pri  7622
mod # 29 item#  1 freq   477  pri  3079
mod # 17 item#  1 freq   516  pri  6300
```

mod	#	58	item#	1	freq	534	pri	2321
mod	#	56	item#	3	freq	569	pri	737
mod	#	7	item#	1	freq	660	pri	3908
mod	#	50	item#	3	freq	672	pri	3498
mod	#	31	item#	1	freq	737	pri	4070
mod	#	18	item#	2	freq	787	pri	2111
mod	#	14	item#	3	freq	894	pri	1811
mod	#	9	item#	2	freq	976	pri	3514
mod	#	49	item#	2	freq	1004	pri	5952
mod	#	25	item#	4	freq	1020	pri	3050
mod	#	5	item#	2	freq	1037	pri	5123
mod	#	14	item#	2	freq	1046	pri	4839
mod	#	10	item#	1	freq	1122	pri	3357
mod	#	5	item#	1	freq	1189	pri	2558
mod	#	25	item#	2	freq	1224	pri	6928
mod	#	47	item#	1	freq	1238	pri	3922
mod	#	53	item#	2	freq	1254	pri	7612
mod	#	57	item#	1	freq	1270	pri	1741
mod	#	22	item#	4	freq	1285	pri	2930
mod	#	25	item#	3	freq	1288	pri	1170
mod	#	25	item#	1	freq	1370	pri	3926
mod	#	10	item#	4	freq	1380	pri	1493
mod	#	53	item#	1	freq	1413	pri	1071
mod	#	16	item#	2	freq	1417	pri	7516
mod	#	29	item#	4	freq	1435	pri	4435
mod	#	13	item#	3	freq	1449	pri	6652
mod	#	29	item#	3	freq	1486	pri	967
mod	#	32	item#	1	freq	1595	pri	2842
mod	#	43	item#	1	freq	1669	pri	7900
mod	#	22	item#	3	freq	1718	pri	2089
mod	#	20	item#	4	freq	1789	pri	3213
mod	#	58	item#	2	freq	1806	pri	6405
mod	#	44	item#	1	freq	1823	pri	2612
mod	#	27	item#	1	freq	1839	pri	7170
mod	#	3	item#	1	freq	1991	pri	6780
mod	#	54	item#	2	freq	1995	pri	1929
mod	#	44	item#	2	freq	2033	pri	6092
mod	#	30	item#	2	freq	2056	pri	2850
mod	#	26	item#	1	freq	2070	pri	2615
mod	#	7	item#	2	freq	2139	pri	6555
mod	#	19	item#	2	freq	2168	pri	7474
mod	#	34	item#	1	freq	2181	pri	6269
mod	#	22	item#	2	freq	2271	pri	5738
mod	#	16	item#	1	freq	2308	pri	2072
mod	#	38	item#	1	freq	2395	pri	3490
mod	#	55	item#	1	freq	2420	pri	6666
mod	#	19	item#	1	freq	2429	pri	5787
mod	#	36	item#	1	freq	2483	pri	3280
mod	#	9	item#	1	freq	2502	pri	4986
mod	#	18	item#	3	freq	2527	pri	5602
mod	#	3	item#	4	freq	2576	pri	5082
mod	#	20	item#	1	freq	2583	pri	2583
mod	#	41	item#	1	freq	2597	pri	4585
mod	#	13	item#	2	freq	2724	pri	6201
mod	#	8	item#	3	freq	2748	pri	618
mod	#	49	item#	1	freq	2755	pri	5034
mod	#	12	item#	1	freq	2771	pri	3736
mod	#	47	item#	3	freq	2796	pri	5692

```
mod #  35 item#  1 freq  2801 pri  7450
mod #   3 item#  2 freq  2825 pri  7900
mod #  30 item#  1 freq  2844 pri   500
mod #  19 item#  4 freq  2979 pri  3182
mod #  20 item#  3 freq  3038 pri  6217
mod #   8 item#  2 freq  3062 pri   500
mod #  46 item#  1 freq  3075 pri  4111
mod #  13 item#  1 freq  3083 pri  1384
mod #  45 item#  3 freq  3091 pri  3460
mod #   8 item#  1 freq  3221 pri  2775
mod #  29 item#  2 freq  3227 pri  7900
mod #  18 item#  1 freq  3250 pri  3044
mod #  50 item#  1 freq  3254 pri  2979
mod #  30 item#  3 freq  3262 pri  3153
mod #  56 item#  1 freq  3279 pri  7189
mod #  21 item#  1 freq  3280 pri  4013
mod #  22 item#  1 freq  3346 pri  3343
mod #  10 item#  3 freq  3383 pri  2862
mod #  58 item#  3 freq  3458 pri  4070
mod #  50 item#  2 freq  3487 pri   766
mod #  45 item#  1 freq  3491 pri  5986
mod #  45 item#  2 freq  3567 pri  6063
mod #  10 item#  5 freq  3614 pri  5007
mod #  56 item#  2 freq  3651 pri  6300
mod #  54 item#  1 freq  3653 pri  1372
mod #  10 item#  2 freq  3696 pri   948
mod #  37 item#  2 freq  3699 pri   803
mod #  14 item#  1 freq  3700 pri  2627
mod #  36 item#  2 freq  3778 pri  2989
mod #  13 item#  4 freq  3796 pri  6243
mod #  47 item#  2 freq  3947 pri  4948
mod #  19 item#  3 freq  3986 pri  5564
mod #  23 item#  1 freq  4057 pri  2008
```

F.4.d. 102 Emitters by PRI

```
mod #  30 item#  1 freq  2844 pri   500
mod #   8 item#  2 freq  3062 pri   500
mod #   8 item#  3 freq  2748 pri   618
mod #  56 item#  3 freq   569 pri   737
mod #  50 item#  2 freq  3487 pri   766
mod #  37 item#  2 freq  3699 pri   803
mod #  10 item#  2 freq  3696 pri   948
mod #  29 item#  3 freq  1486 pri   967
mod #  53 item#  1 freq  1413 pri  1071
mod #  25 item#  3 freq  1288 pri  1170
mod #  54 item#  1 freq  3653 pri  1372
mod #  13 item#  1 freq  3083 pri  1384
mod #  10 item#  4 freq  1380 pri  1493
mod #  57 item#  1 freq  1270 pri  1741
mod #  14 item#  3 freq   894 pri  1811
mod #  54 item#  2 freq  1995 pri  1929
mod #  23 item#  1 freq  4057 pri  2008
mod #  16 item#  1 freq  2308 pri  2072
mod #  22 item#  3 freq  1718 pri  2089
mod #  18 item#  2 freq   787 pri  2111
mod #  58 item#  1 freq   534 pri  2321
```

mod #	5	item#	1	freq	1189	pri	2558
mod #	20	item#	1	freq	2583	pri	2583
mod #	44	item#	1	freq	1823	pri	2612
mod #	26	item#	1	freq	2070	pri	2615
mod #	14	item#	1	freq	3700	pri	2627
mod #	8	item#	1	freq	3221	pri	2775
mod #	32	item#	1	freq	1595	pri	2842
mod #	30	item#	2	freq	2056	pri	2850
mod #	10	item#	3	freq	3383	pri	2862
mod #	22	item#	4	freq	1285	pri	2930
mod #	37	item#	1	freq	253	pri	2975
mod #	50	item#	1	freq	3254	pri	2979
mod #	36	item#	2	freq	3778	pri	2989
mod #	29	item#	1	freq	477	pri	3079
mod #	30	item#	3	freq	3262	pri	3153
mod #	19	item#	4	freq	2979	pri	3182
mod #	20	item#	4	freq	1789	pri	3213
mod #	36	item#	1	freq	2483	pri	3280
mod #	22	item#	1	freq	3346	pri	3343
mod #	10	item#	1	freq	1122	pri	3357
mod #	45	item#	3	freq	3091	pri	3460
mod #	38	item#	1	freq	2395	pri	3490
mod #	50	item#	3	freq	672	pri	3498
mod #	9	item#	2	freq	976	pri	3514
mod #	12	item#	1	freq	2771	pri	3736
mod #	7	item#	1	freq	660	pri	3908
mod #	47	item#	1	freq	1238	pri	3922
mod #	25	item#	1	freq	1370	pri	3926
mod #	21	item#	1	freq	3280	pri	4013
mod #	58	item#	3	freq	3458	pri	4070
mod #	31	item#	1	freq	737	pri	4070
mod #	46	item#	1	freq	3075	pri	4111
mod #	29	item#	4	freq	1435	pri	4435
mod #	41	item#	1	freq	2597	pri	4585
mod #	14	item#	2	freq	1046	pri	4839
mod #	47	item#	2	freq	3947	pri	4948
mod #	9	item#	1	freq	2502	pri	4986
mod #	10	item#	5	freq	3614	pri	5007
mod #	49	item#	1	freq	2755	pri	5034
mod #	31	item#	2	freq	248	pri	5079
mod #	3	item#	4	freq	2576	pri	5082
mod #	5	item#	2	freq	1037	pri	5123
mod #	4	item#	1	freq	167	pri	5267
mod #	56	item#	4	freq	26	pri	5476
mod #	19	item#	3	freq	3986	pri	5564
mod #	18	item#	3	freq	2527	pri	5602
mod #	47	item#	3	freq	2796	pri	5692
mod #	22	item#	2	freq	2271	pri	5738
mod #	19	item#	1	freq	2429	pri	5787
mod #	49	item#	2	freq	1004	pri	5952
mod #	45	item#	1	freq	3491	pri	5986
mod #	35	item#	2	freq	6	pri	6025
mod #	45	item#	2	freq	3567	pri	6063
mod #	44	item#	2	freq	2033	pri	6092
mod #	7	item#	3	freq	280	pri	6103
mod #	13	item#	2	freq	2724	pri	6201

mod #	20	item#	3	freq	3038	pri	6217
mod #	13	item#	4	freq	3796	pri	6243
mod #	34	item#	1	freq	2181	pri	6269
mod #	56	item#	2	freq	3651	pri	6300
mod #	17	item#	1	freq	516	pri	6300
mod #	58	item#	2	freq	1806	pri	6405
mod #	7	item#	2	freq	2139	pri	6555
mod #	13	item#	3	freq	1449	pri	6652
mod #	55	item#	1	freq	2420	pri	6666
mod #	3	item#	1	freq	1991	pri	6780
mod #	25	item#	2	freq	1224	pri	6928
mod #	3	item#	3	freq	282	pri	7096
mod #	27	item#	1	freq	1839	pri	7170
mod #	56	item#	1	freq	3279	pri	7189
mod #	35	item#	1	freq	2801	pri	7450
mod #	19	item#	2	freq	2168	pri	7474
mod #	16	item#	2	freq	1417	pri	7516
mod #	53	item#	2	freq	1264	pri	7612
mod #	20	item#	2	freq	410	pri	7622
mod #	3	item#	2	freq	2825	pri	7900
mod #	43	item#	1	freq	1669	pri	7900
mod #	29	item#	2	freq	3227	pri	7900

APPENDIX F.5. 300 EMITTERS WITH STATIONARY ENVIRONMENT

Following are the simulation results from an environment with 300 emitters in which the velocities of the airborne platform and all emitters are set, artificially, to zero to determine the effect of DOA drifts to memory module 3, as discussed in section D.6.a. For compactness, the results are only shown at 1, 5, 10, 15, and 20 second intervals. System parameters are as follows:

maximum on-time separation/100 emitters in seconds	.05
Associative Processor time in microseconds	1.0
Array-Processing time/microinstruction in microsecs.	0.1
CAM-manager processing time in microseconds	1.0
number of CAM registers	24

```
TIME =  1.00 SEC.    (PRINT INCR=  1.000)

NMCNT=       566  MCNT=     96397  MAX=     243  CAMB=  5  MPPB=  3
 MTIME= 0.035659  NCWL=     96966  MLCAMBUF=  3  IDOAD=      1
 ICFD=     1  NEA=     97941  NIA=    1543  INLT1=    0  INLT2=    1  INLT3=  864
   0  2  6  4  1  4  2  6  8 12  4  4  5 10  2  9
   5 11  7  7  5  6  6  4  8  2  4  3  4  5  5  5
   6  3  3  8  5  4  2  0  5  2  6  5  7  7  7  4
   9  5  5  8  5  3  7  4  7  3  1  1  1  0  0  0
MAIN MEMORY SIZES:
   0  2  6  4  1  4  2  6  8 12  4  4  5 10  2  9
   5 11  7  7  5  6  6  4  8  2  4  3  4  5  5  5
   6  3  3  8  5  4  2  0  5  2  6  5  7  7  7  3
   9  5  5  8  5  3  7  4  7  3  1  1  1  0  0  0
    TOTAL EMITERS IN MPPR=298
  14 12 12 15 13 14 14 15 13 12 15 15 17 12 13 14
  13 12 13 15 15 12 11 15 11 15 13 12 12 13 15 15
  14 12 15 14 15 13 12 15 14 12 13 14 13 13 13 13
  15 15 14 13 13 15 13 15 15 13 14 14 14 14 14 18
  12 13 16 13 15 14 12 12 15 13 13 14 15 15 13 12
  14 13 13 13 14 13 12 16 15 15 15 13 13 17 12 14
  14 14 12 15 15 13 14 14 15 14 15 14 14 14 15 14
  15 14 12 14 12 14 12 14 13 16 15 12 13 15 14 13
LIST SIZES:
   0  0  0  0  0  0  0  0  0  7 10  4  6 10  5  7
   6  7  4  4  6  7  9  0  1  5  8  7  4  5  6  5
   2  2 10  4  2  3  9  2  8  4  6  6  3  2  4  2
   2  0  4  4  2  5  2  0  2  2  1  2  1  5  1  3
   1  1  2  2  2  2  0  5  0  3  0  4  0  3  4  1
   1  0  0  0  0  1  1  0  0  0  2  1  2  0  1  0
   1  1  0  2  1  2  0  2  1  1  1  0  1  0  0  3
   1  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0
 TOTAL CONTENTS= 296    AVERAGE CONTENTS=  2.31
```

```
TIME =  5 00 SEC     (PRINT INCR=  1.000)

NMCNT=      3152  MCNT=    516195  MAX=     243  CAMB=  5  MPPB=  3
 MTIME= 0.035659  NCWL=    519347  MLCAMBUF=  4  IDOAD=      1
 ICFD=      1  NEA=    520325  NIA=     4129  INLT1=    0  INLT2=    1  INLT3= 3450
   0  2  6  4  1  4  2  6  8 12  4  4  5 10  2  9
   5 11  7  7  5  6  6  4  8  2  4  3  4  5  5  5
   6  3  3  8  5  4  2  0  5  2  6  5  7  7  7  4
   9  5  5  8  5  3  7  4  7  3  1  1  1  0  0  0
MAIN MEMORY SIZES
   0  2  6  4  1  4  2  6  8 12  4  4  5 10  2  9
   5 11  7  7  5  6  6  4  8  2  4  3  4  5  5  5
   6  3  3  8  5  4  2  0  5  2  6  5  7  7  7  3
   9  5  5  8  5  3  7  4  7  3  1  1  1  0  0  0
    TOTAL EMITERS IN MPPH=298
  16 14 15 17 18 16 18 15 17 16 15 16 17 17 14 16
  17 15 17 15 17 15 14 15 14 16 15 13 14 15 17 15
  21 15 15 17 15 15 15 17 18 16 17 17 16 16 14 15
  15 15 16 16 14 15 18 16 15 15 16 15 17 17 15 18
  17 15 16 16 17 16 14 16 16 17 16 20 16 18 16 15
  15 16 14 16 16 16 16 16 17 16 15 15 14 17 16 16
  16 16 15 15 17 15 15 16 16 15 16 15 15 16 16 17
  15 15 15 17 16 14 18 16 16 16 16 15 15 15 14 14
LIST SIZES:
   2  1  0  1  0  0  0  1  1  1  0  0  1  0  0  1
   0  0  0  1  0  0  2  0  0  0  1  2  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  0 12  4  5
   5  7  5  6  5  6  9  8  7  5  6 10  7  1  6  8
   1  7  3  5  4  2  2  4  1  3  4  2  3  5  2  4
   4  4  6  6  2  2  2  5  1  4  2  2  4  4  2  2
   4  1  2  1  0  0  0  2  3  2  1  2  3  1  4  6
   4  1  3  2  1  2  2  3  1  0  1  1  1  2  0  1
TOTAL CONTENTS= 299     AVERAGE CONTENTS=  2.34


TIME =  10.00 SEC.    (PRINT INCR=  1.000)

NMCNT=      6570  MCNT=   1040764  MAX=     243  CAMB=  5  MPPB=  3
 MTIME= 0.035659  NCWL=   1047334  MLCAMBUF=  4  IDOAD=      1
 ICFD=      1  NEA=   1048312  NIA=     7547  INLT1=    0  INLT2=    1  INLT3= 6968
   0  2  6  4  1  4  2  6  8 12  4  4  5 10  2  9
   5 11  7  7  5  6  6  4  8  2  4  3  4  5  5  5
   6  3  3  8  5  4  2  0  5  2  6  5  7  7  7  4
   9  5  5  8  5  3  7  4  7  3  1  1  1  0  0  0
MAIN MEMORY SIZES:
   0  2  6  4  1  4  2  6  8 12  4  4  5 10  2  9
   5 11  7  7  5  6  6  4  8  2  4  3  4  5  5  5
   6  3  3  8  5  4  2  0  5  2  6  5  7  7  7  3
   9  5  5  8  5  3  7  4  7  3  1  1  1  0  0  0
    TOTAL EMITERS IN MPPR=298
  16 15 19 17 18 16 18 17 17 17 16 16 17 17 15 16
  17 15 17 17 21 19 14 17 17 18 16 14 15 16 18 17
  21 15 16 17 16 17 17 17 18 16 17 17 17 16 16 17
  16 17 16 16 15 15 18 17 16 14 16 15 17 17 16 18
  17 15 16 17 17 16 15 17 16 17 16 20 16 18 17 15
  15 16 16 16 18 16 16 16 19 17 16 17 17 17 16 16
  16 16 15 18 17 15 18 16 16 17 16 15 15 16 17 17
  17 15 17 17 16 15 18 16 17 16 17 15 17 15 15 16
 LIST SIZES:
   4  4  5  2  4  3  4  4  2  1  0  1  5  2  5  0
   4  2  0  1  3  2  3  0  3  2  3  0  1  0  1  3
   0  4  1  3  3  2  2  1  2  0  1  5  2  3  0  1
   0  1  1  2  0  0  0  0  0  1  1  1  0  0  0  0
   0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  4  6  5  3  4  9  9
   8  5  8  5 13  6  5  3  6  8  4  6  6  2  4  4
   7  2  2  4  2  6  5  6  3  2  2  1  3  4  4  2
 TOTAL CONTENTS= 299     AVERAGE CONTENTS=  2.34


 TIME =  11.00 SEC.    (PRINT INCR=  1.000)
```

```
TIME = 15.00 SEC.    (PRINT INCR= 1.000)

NMCNT=     10010  MCNT=  1565344  MAX=     243  CAMB=  5  MPPB=  4
 MTIME= 0.035659  NCWL=  1575361  MLCAMBUF=  5  IDOAD=      5
 ICFD=     5  NEA=  1576336  NIA=  10991  INLT1=    0  INLT2=    5  INLT3=10000
   0  2  6  4  1  4  2  6  8 12  4  4  5 10  2  9
   5 11  7  7  5  6  6  4  8  2  4  3  4  5  5  5
   6  3  3  8  5  4  2  0  5  2  6  5  7  7  7  4
   9  5  5  8  5  3  7  4  7  3  1  1  1  0  0  0
MAIN MEMORY SIZES.
   0  2  6  4  1  4  2  6  8 12  4  4  5 10  2  9
   5 11  7  7  5  6  6  4  8  2  4  3  4  5  5  5
   6  3  3  8  5  4  2  0  5  2  6  5  7  7  7  3
   9  5  5  8  5  3  7  4  7  3  1  1  1  0  0  0
    TOTAL EMITERS IN MPPR=298
  16 16 19 17 19 17 18 17 17 17 16 17 17 17 16 16
  17 18 17 17 21 19 15 17 17 18 16 16 15 16 18 17
  21 16 16 17 16 17 17 17 18 18 17 17 17 17 17 17
  16 17 17 17 17 18 18 17 16 16 17 15 17 17 16 18
  17 15 16 17 17 16 15 17 16 17 16 20 16 18 17 15
  15 17 16 16 18 16 16 16 19 17 16 18 17 17 16 17
  16 16 18 18 17 16 18 16 16 17 17 16 15 16 17 17
  17 15 17 17 16 16 18 16 17 18 17 16 17 15 15 16
LIST SIZES
   0  0  0  0  0  0  0  8  6  4  7  7  4  8  6  7
   3  7  6 10  7  5  7  2  5  5  6  5  4  1  5  6
   2  8  1  2  5  4  3  2  6  1  3  5  2  4  3  5
   2  2  2  4  1  1  2  6  3  2  5  3  3  4  0  2
   0  3  1  1  4  0  0  0  3  1  4  3  4  3  4  1
   2  2  1  2  2  1  0  0  2  0  0  1  0  2  0  1
   0  0  0  0  1  0  0  2  0  0  2  1  0  0  1  0
   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
TOTAL CONTENTS= 296    AVERAGE CONTENTS=  2.31
```

```
TIME = 20.00 SEC.    (PRINT INCR= 1.000)

NMCNT=     14066  MCNT=  2089272  MAX=     243  CAMB=  5  MPPB=  6
 MTIME= 0.035659  NCWL=  2103343  MLCAMBUF=  7  IDOAD=      7
 ICFD=     7  NEA=  2104322  NIA=  15049  INLT1=    0  INLT2=    7  INLT3=14364
   0  2  6  4  1  4  2  6  8 12  4  4  5 10  2  9
   5 11  7  7  5  6  6  4  8  2  4  3  4  5  5  5
   6  3  3  8  5  4  2  0  5  2  6  5  7  7  7  4
   9  5  5  8  5  3  7  4  7  3  1  1  1  0  0  0
MAIN MEMORY SIZES:
   0  2  6  4  1  4  2  6  8 12  4  4  5 10  2  9
   5 11  7  7  5  6  6  4  8  2  4  3  4  5  5  5
   6  3  3  8  5  4  2  0  5  2  6  5  7  7  7  3
   9  5  5  8  5  3  7  4  7  3  1  1  1  0  0  0
    TOTAL EMITERS IN MPPR=298
  16 16 19 17 19 20 18 17 17 17 16 17 17 18 16 16
  17 18 17 17 21 19 16 17 17 18 16 16 15 16 18 17
  21 17 16 17 16 17 17 19 18 18 17 17 17 17 16 17
  16 17 17 17 17 19 18 17 19 16 17 15 17 17 16 19
  17 15 18 17 17 16 19 17 16 17 17 20 16 18 17 15
  15 17 16 16 18 16 16 16 19 17 16 18 17 17 16 18
  17 16 18 18 17 16 18 16 18 17 17 16 16 16 17 17
  17 17 17 17 16 16 18 16 17 18 17 16 17 15 16 18
 LIST SIZES:
   0  3  1  0  2  0  1  2  1  0  1  0  0  1  0  1
   2  0  0  1  0  1  0  2  1  1  0  0  0  1  0  0
   0  0  0  0  1  1  2  0  0  0  0  0  0  0  0  0
   0  0  0  4  8  6  4  8  9  6  7  5  9  5  7  6
   3  8  9  5  2  5  5  9  5  3  3  4  2  4  4  4
   5  3  1  3  3  2  4  5  3  4  3  6  4  2  1  2
   0  4  1  4  1  5  4  2  1  1  2  4  1  4  0  4
   2  0  4  0  0  2  0  3  1  2  2  0  2  4  1  1
 TOTAL CONTENTS= 300    AVERAGE CONTENTS=  2.34
```

APPENDIX F.6. FREQUENCY HOPPER ENVIRONMENT

Following are the run results two environments in which the number of emitters in the environment was 50, which included two frequency hoppers. In the first case, in order to accommodate the overflow in the main memory modules which corresponded to the frequency hoppers, new data was entered into the top of the module when the module became full; in other words, the data wraps around and writes over the previously entered data. In the second case, when the memory module is filled, no more data is loaded into that module. For more information, see section D.1.b.

The simulation was run with a static LIST of (32,256) for one second. The results shown here are for the entire run with no wraparound, but only from 0.5 seconds for wraparound memory. The results were identical up to that time.

sFH: Simulation Results for 50 Emitters
and 2 Frequency Hoppers
Wraparound Memory

```
TIME =  0.10 SEC.    (PRINT INCR=  0.100)

NMCNT=         0  MCNT=      1105  MAX=     200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=      1111  MLCAMBUF=  2  IDOAD=      1
 ICFD=     0  NEA=      1298  NIA=      192  INLT1=    0  INLT2=    2  INLT3=   47
   0  0  0. 1  1  1  2  2  0  1  0  2  1  1  0 13
   3  2  1  0  0  1  0  1  1  0  0  3  0  0  0  0
   0 31  1  1  0  0  0  2  1  1  0  1  1  3  3  1
   0  2  0  0  1  1  2  0  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES:
   0  0  0  0  1  1  1  2  0  1  0  2  1  1  0 13
   3  2  1  0  0  1  0  1  1  0  0  3  0  0  0  0
   0 31  1  1  0  0  0  2  1  1  0  1  1  3  3  1
   0  2  0  0  1  1  2  0  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR= 88
   8  7  6  6  6  8  6  8  6  6  7  6 11  5  5  8
   6  6  6  5  6  7  6  7  7  9  7  6  6  6  9  7
LIST SIZES:
   0  0  1  0  0  0  0  2  4  7  4  3  4  0  3  2
   1  1  2  2  0  1  2  0  0  1  0  2  1  0  0  0
TOTAL CONTENTS=  43    AVERAGE CONTENTS=  1.34
```

```
TIME =  0 20 SEC     (PRINT INCR=  0.100)

NMCNT=         1  MCNT=      2720  MAX=     200  CAMB=  3  MPPB=  2
 MTIME= 0 059462  NCWL=      2735  MLCAMBUF=  2  IDOAD=       1
 ICFD=     0  NEA=      2971  NIA=      240  INLT1=    0  INLT2=    2  INLT3=    48
   0  0  0  1  1  1  2  2  0  1  0  2  1  1  0 25
   3  2  1  0  0  1  0  1  1  0  0  3  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  1
   0  2  0  0  1  1  2  0  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES
   0  0  0  1  1  1  1  2  0  1  0  2  1  1  0 25
   3  2  1  0  0  1  0  1  1  0  0  3  0  0  0  0
   0 16  1  1  0  0  0  2  1  1  0  1  1  3  3  1
   0  2  0  0  1  1  2  0  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR= 85  (INT)
   8  7  8  7  7  8  8  8  8  7  7  7 11  9  7  8
   8 11  6  8  7  8  6  7  8  9  7  6  9  8  9  8
LIST SIZES.
   2  1  1  2  0  0  0  0  0  0  0  0  0  4  4  2
   3  5  4  1  1  1  3  2  1  2  1  1  1  1  2  0
TOTAL CONTENTS=  45    AVERAGE CONTENTS=  1.41


TIME =  0.30 SEC.    (PRINT INCR=  0.100)

NMCNT=         1  MCNT=      4351  MAX=     200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=      4356  MLCAMBUF=  2  IDOAD=       3
 ICFD=     0  NEA=      4641  NIA=      289  INLT1=    0  INLT2=    4  INLT3=    48
   0  0  0  1  1  1  2  2  0  1  0  2  1  1  0 38
   3  2  1  0  0  1  0  2  1  0  0  3  0  0  0  0

   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  2
   0  2  0  0  1  1  2  0  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES.
   0  0  0  1  1  1  1  2  0  1  0  2  1  1  0 35
   3  2  1  0  0  1  0  2  0  0  0  3  0  0  0  0
   0  2  1  1  0  0  0  2  1  1  0  1  1  3  2  2
   0  2  0  0  1  1  2  0  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR= 84 (130)
   8  8  8  8  7  8  8  8  8  7  7  7 11  9  8  8
   8 11  6  8  9  8  8  7  8  9  9  9  9  9  9  8
LIST SIZES:
   1  0  2  0  2  1  1  0  0  3  0  0  1  0  0  0
   0  0  0  3  5  4  3  1  3  5  1  3  3  2  2  1
TOTAL CONTENTS=  47    AVERAGE CONTENTS=  1 47


TIME =  0.40 SEC.    (PRINT INCR=  0.100)

NMCNT=         1  MCNT=      5972  MAX=     200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=      5984  MLCAMBUF=  2  IDOAD=       7
 ICFD=     0  NEA=      6313  NIA=      340  INLT1=    0  INLT2=    8  INLT3=    48
   0  0  0  1  1  1  2  2  0  1  1  2  2  1  0 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  2
   0  2  0  0  1  1  2  1  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES.
   0  0  0  1  1  1  1  2  0  1  1  1  2  0  0  3
   3  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0 36  1  1  0  0  0  2  1  1  0  1  1  3  2  2
   0  2  0  0  1  1  1  1  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR= 83 (140)
   8  8  8  8  7  8  9  8  8  9  8  7 11  9  8  8
   8 11  6  8  9  8  8  8  8  9  9  9  9  9  9  8
LIST SIZES:
   2  2  3  0  3  0  4  0  1  0  0  0  1  2  0  0
   1  1  1  0  0  0  0  0  0  0  2  2  6  5  5  1
TOTAL CONTENTS=  44    AVERAGE CONTENTS=  1.38
```

```
TIME =  0.50 SEC     (PRINT INCR=  0.100)

NMCNT=         1  MCNT=      7604  MAX=     200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=      7615  MLCAMBUF=  2  IDOAD=      8
 ICFD=     0  NEA=      7992  NIA=       [illegible]  INLT1=   0  INLT2=   9  INLT3=   48
   0  0  0  1  1  1  2  2  0  1  1  2  2  1  0 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  2
   0  2  0  0  1  1  2  1  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES
   0  0  0  1  1  1  1  2  0  1  1  1  2  0  0 16
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0 20  1  1  0  0  0  2  1  1  0  1  1  3  2  2
   0  2  0  0  1  1  1  1  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR= 79     (139)
   8  8  8  8  7  8  9  8  8  9  8  9 11  9  8  8
   8 11  7  9  9  8  8  9  9  9  9  9  9  9  9  9
LIST SIZES
   0  2  2  3  2  2  2  3  1  3  3  7  2  3  2  2
   0  3  1  1  0  0  0  0  1  0  0  0  1  0  0  0


TOTAL CONTENTS=  46    AVERAGE CONTENTS=  1.44


TIME =  0.60 SEC     (PRINT INCR=  0.100)

NMCNT=         1  MCNT=      9216  MAX=     200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=      9231  MLCAMBUF=  2  IDOAD=     11
 ICFD=     0  NEA=      9661  NIA=       444  INLT1=   0  INLT2=  12  INLT3=   50
   0  0  0  2  1  1  2  2  0  1  1  2  2  1  2 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  2
   1  2  0  0  1  1  2  1  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES.
   0  0  0  2  0  1  2  1  0  1  1  1  2  0  2 29
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0  6  1  1  0  0  0  2  1  1  0  1  1  3  2  1
   1  2  0  0  1  1  1  1  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR= 80  (141)
   8  8  8  8  7  8  9  8  8  9  8 11 11  9  8  8
   8 11  8  9  9  8  8  9  9  9  9  9  9  9  9  9
LIST SIZES:
   0  1  0  0  0  0  0  5  5  2  1  3  4  3  2  4
   1  1  2  0  3  1  2  1  0  1  0  1  1  2  0  1
TOTAL CONTENTS=  47    AVERAGE CONTENTS=  1.47


TIME =  0.70 SEC.    (PRINT INCR=  0.100)

NMCNT=         1  MCNT=     10841  MAX=     200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=     10857  MLCAMBUF=  2  IDOAD=     12
 ICFD=     0  NEA=     11334  NIA=       492  INLT1=   0  INLT2=  13  INLT3=   50
   0  0  0  2  1  1  2  2  0  1  1  2  2  1  2 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  2
   1  2  0  0  1  1  2  1  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES
   0  0  0  2  0  1  2  1  0  1  1  1  2  0  2 42
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0 38  1  1  0  0  0  2  1  1  0  1  1  2  3  1
   1  2  0  0  1  1  1  1  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR=123    (141)
   8  9  8  8  7  8  9  8  8  9  8 11 11  9  8  8
   9 11  8 11  9  8  9  9  9  9  9  9  9  9 11  9
LIST SIZES.
   1  1  2  0  0  0  2  0  1  0  0  0  0  0  6  6
   3  3  2  3  1  2  2  0  2  1  4  1  1  2  1  0
TOTAL CONTENTS=  47    AVERAGE CONTENTS=  1.47
```

```
TIME =  0.80 SEC     (PRINT INCR=  0.100)

NMCNT=         1  MCNT=     12461  MAX=     200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=     12478  MLCAMBUF=   2  IDOAD=     13
 ICFD=      0  NEA=      13001  NIA=       539  INLT1=    0  INLT2=  14  INLT3=   50
   0  0  0  2  1  1  2  2  0  1  1  2  2  1  7 48
   3  2  1  0  0  1  0  2  1  0  1  0  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  2  2  2

   1  2  0  0  1  1  2  2  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES
   0  0  0  2  0  1  2  1  0  1  1  1  2  0  7  0
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0 22  1  1  0  0  0  2  1  1  0  1  1  2  3  1
   1  2  0  0  1  1  0  2  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR= 72  (149)
   9  9  8  9  9  8  9  8  8  9  8 11 11  9  8  8
   8 11  8 11  9  8  9  9  9  9 10  9  9  9 11  9
LIST SIZES:
   3  1  0  0  2  3  1  0  0  0  0  0  0  0  1  0
   0  0  0  0  4  2  3  4  4  3  2  2  1  1  1  4
TOTAL CONTENTS=  47    AVERAGE CONTENTS=  1.47


TIME =  0.90 SEC     (PRINT INCR=  0.100)

NMCNT=         1  MCNT=     14088.  MAX=     200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=     14106  MLCAMBUF=   2  IDOAD=     14
 ICFD=      0  NEA=      14677  NIA=       588  INLT1=    0  INLT2=  15  INLT3=   50
   0  0  0  2  1  1  2  2  0  1  1  2  2  1 18 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  2
   1  2  1  0  1  1  2  2  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES:
   0  0  0  2  0  1  2  1  0  1  1  1  2  0 18  0
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0  8  1  1  0  0  0  2  1  1  0  1  1  2  3  1
   1  1  1  0  1  1  0  2  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR= 69  (157)
   8  9  8  9  9  8  9  8  9  9  8 11 11  9  8  8
   8 11  8 11  9  8  9  9  9  9 10  9  9  9 11  9
LIST SIZES:
   4  6  2  3  3  5  1  0  0  0  1  1  0  1  2  0
   0  1  1  0  0  1  0  0  0  0  2  1  3  2  1  3
TOTAL CONTENTS=  47    AVERAGE CONTENTS=  1.47


TIME =  1.00 SEC.    (PRINT INCR=  0.100)

NMCNT=         1  MCNT=     15708  MAX=     200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=     15731  MLCAMBUF=   2  IDOAD=     17
 ICFD=      0  NEA=      16350  NIA=       641  INLT1=    0  INLT2=  18  INLT3=   51
   0  0  0  2  1  1  2  2  0  1  1  2  2  1 31 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  2  1  0  0  0  2  1  2  1  1  1  3  3  2
   1  2  1  0  1  1  2  2  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES.
   0  0  0  2  0  1  2  1  0  1  1  1  2  0 31 47
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0 38  2  1  0  0  0  2  0  1  1  1  1  2  3  1
   1  1  1  0  1  1  0  2  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR=159  (170)
   9  9  8 10  9  9 11  9  9 10  8 11 11  9  8 11
   8 11  8 11  9  8  9  9  9  9 10  9  9  9 11 10
LIST SIZES:
   0  0 11  2  1  3  0  3  3  0  1  2  2  1  1  2
   2  1  2  0  0  0  0  0  3  1  0  1  0  0  0  0
TOTAL CONTENTS=  46    AVERAGE CONTENTS=  1.44
```

F.6.4

Simulation Results for 50 Emitters and 2 Frequency Hoppers No Wraparound in Memory

```
TIME =  0.50 SEC.   (PRINT INCR=  0.100)

NMCNT=        1  MCNT=     7592  MAX=    200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=     7603  MLCAMBUF=  2  IDOAD=      8
 ICFD=     0  NEA=     7992  NIA=    399  INLT1=   0  INLT2=   8  INLT3=   48
   0  0  0  1  1  1  2  2  0  1  1  2  2  1  0 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  2
   0  2  0  0  1  1  2  1  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES:
   0  0  0  1  1  1  1  2  0  1  1  1  2  0  0 48
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  2  2
   0  2  0  0  1  1  1  1  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR=139
   8  8  8  8  7  8  9  8  8  9  8  9 11  9  8  8
   8 11  7  9  9  8  8  9  9  9  9  9  9  9  9  8
LIST SIZES:
   0  2  2  3  2  2  2  3  1  3  3  6  2  3  2  2
   0  3  1  1  0  0  0  0  1  0  0  0  1  0  0  0
TOTAL LIST SIZE=  45    AVG LIST SIZE=  1.41


TIME =  0.60 SEC.   (PRINT INCR=  0.100)

NMCNT=        2  MCNT=     9200  MAX=    200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=     9216  MLCAMBUF=  2  IDOAD=     13
 ICFD=     0  NEA=     9661  NIA=    460  INLT1=   0  INLT2=  13  INLT3=   50
   0  0  0  2  1  1  2  2  0  1  1  2  2  1  2 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  2
   1  2  0  0  1  1  2  1  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES:
   0  0  0  2  0  1  2  1  0  1  1  1  2  0  2 48
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  2  1
   1  2  0  0  1  1  1  1  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR=141
   8  8  8  8  7  8  9  8  8  9  8 11 11  9  8  8
   8 11  8  9  9  8  8  9  9 10  9  9  9  9  9  8
LIST SIZES:
   0  1  0  0  0  0  0  5  5  2  1  3  4  3  2  4
   1  1  2  0  3  1  2  1  0  1  0  1  1  2  0  1
TOTAL LIST SIZE=  47    AVG LIST SIZE=  1.47



TIME =  0.70 SEC.   (PRINT INCR=  0.100)

NMCNT=        2  MCNT=    10825  MAX=    200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=    10842  MLCAMBUF=  2  IDOAD=     14
 ICFD=     0  NEA=    11334  NIA=    508  INLT1=   0  INLT2=  14  INLT3=   50
   0  0  0  2  1  1  2  2  0  1  1  2  2  1  2 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  2
   1  2  0  0  1  1  2  1  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES:
   0  0  0  2  0  1  2  1  0  1  1  1  2  0  2 48
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  2  3  1
   1  2  0  0  1  1  1  1  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR=141
   8  9  8  8  7  8  9  8  8  9  8 11 11  9  8  8
   8 11  8 11  9  8  9  9  9 10  9  9  9  9 11  8
LIST SIZES:
   1  1  2  0  0  0  2  0  1  0  0  0  0  0  6  6
   3  3  2  3  1  2  2  0  2  1  4  1  1  2  1  0
TOTAL LIST SIZE=  47    AVG LIST SIZE=  1.47
```

```
TIME =  0.80 SEC.   (PRINT INCR=  0.100)

NMCNT=        2  MCNT=    12445  MAX=    200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=    12463  MLCAMBUF=  2  IDOAD=     15
 ICFD=     0  NEA=    13001  NIA=    555  INLT1=    0  INLT2=  15  INLT3=   50
   0  0  0  2  1  1  2  2  0  1  1  2  2  1  8 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  2
   1  2  0  0  1  1  2  2  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES:
   0  0  0  2  0  1  2  1  0  1  1  1  2  0  8 48
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  2  3  1
   1  2  0  0  1  1  0  2  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR=147
   8  9  8  9  9  8  9  8  8  9  8 11 11  9  8  8
   8 11  8 11  9  8  9  9  9 10 10  9  9  9 11  8
LIST SIZES:
   3  1  0  0  2  3  1  0  0  0  0  0  0  0  1  0
   0  0  0  0  4  2  8  4  4  3  2  2  1  1  1  4
TOTAL LIST SIZE=  47    AVG LIST SIZE=  1.47



TIME =  0.90 SEC.   (PRINT INCR=  0.100)

NMCNT=        2  MCNT=    14072  MAX=    200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=    14092  MLCAMBUF=  2  IDOAD=     16
 ICFD=     1  NEA=    14677  NIA=    604  INLT1=    0  INLT2=  16  INLT3=   51
   0  0  0  2  1  1  2  2  0  1  1  2  2  1 20 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  3  3  2
   1  2  1  0  1  1  2  2  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES:
   0  0  0  2  0  1  2  1  0  1  1  1  2  0 20 48
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0 48  1  1  0  0  0  2  1  1  0  1  1  2  3  1
   1  1  1  0  1  1  0  2  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR=159
   8  9  8  9  9  8  9  8  9  9  8 11 11  9  8  8
   8 11  8 11  9  8  9  9  9 10 10  9  9  9 11  8
LIST SIZES:
   4  6  2  3  3  5  1  0  0  0  1  1  0  1  2  0
   0  1  1  0  0  1  0  0  0  0  2  1  6  2  1  3
TOTAL LIST SIZE=  47    AVG LIST SIZE=  1.47



TIME =  1.00 SEC.   (PRINT INCR=  0.100)

NMCNT=        2  MCNT=    15694  MAX=    200  CAMB=  3  MPPB=  2
 MTIME= 0.059462  NCWL=    15719  MLCAMBUF=  2  IDOAD=     20
 ICFD=     1  NEA=    16350  NIA=    655  INLT1=    0  INLT2=  20  INLT3=   51
   0  0  0  2  1  1  2  2  0  1  1  2  2  1 34 48
   3  2  1  0  0  1  0  2  1  0  1  3  0  0  0  0
   0 48  2  1  0  0  0  2  1  2  1  1  1  3  3  2
   1  2  1  0  1  1  2  2  0  0  0  0  0  0  0  0
MAIN MEMORY SIZES:
   0  0  0  2  0  1  2  1  0  1  1  1  2  0 34 47
   2  2  1  0  0  1  0  2  0  0  1  2  0  0  0  0
   0 48  2  1  0  0  0  2  0  1  1  1  1  2  3  1
   1  1  1  0  1  1  0  2  0  0  0  0  0  0  0  0
    TOTAL EMITERS IN MPPR=173
   9  8 10  9  8 11  8  9 10  8 11 11  9  8 11
   8 11  8 11  9  8  9  9  9 10 10  9  9 10 11 10
LIST SIZES:
   0  0 11  2  1  3  0  5  5  0  1  2  2  1  1  2
   2  1  2  0  0  0  0  0  3  1  0  1  0  0  0  0
TOTAL LIST SIZE=  46    AVG LIST SIZE=  1.44
```

F.6.6

APPENDIX F.7. Program Listing and Flowcharts

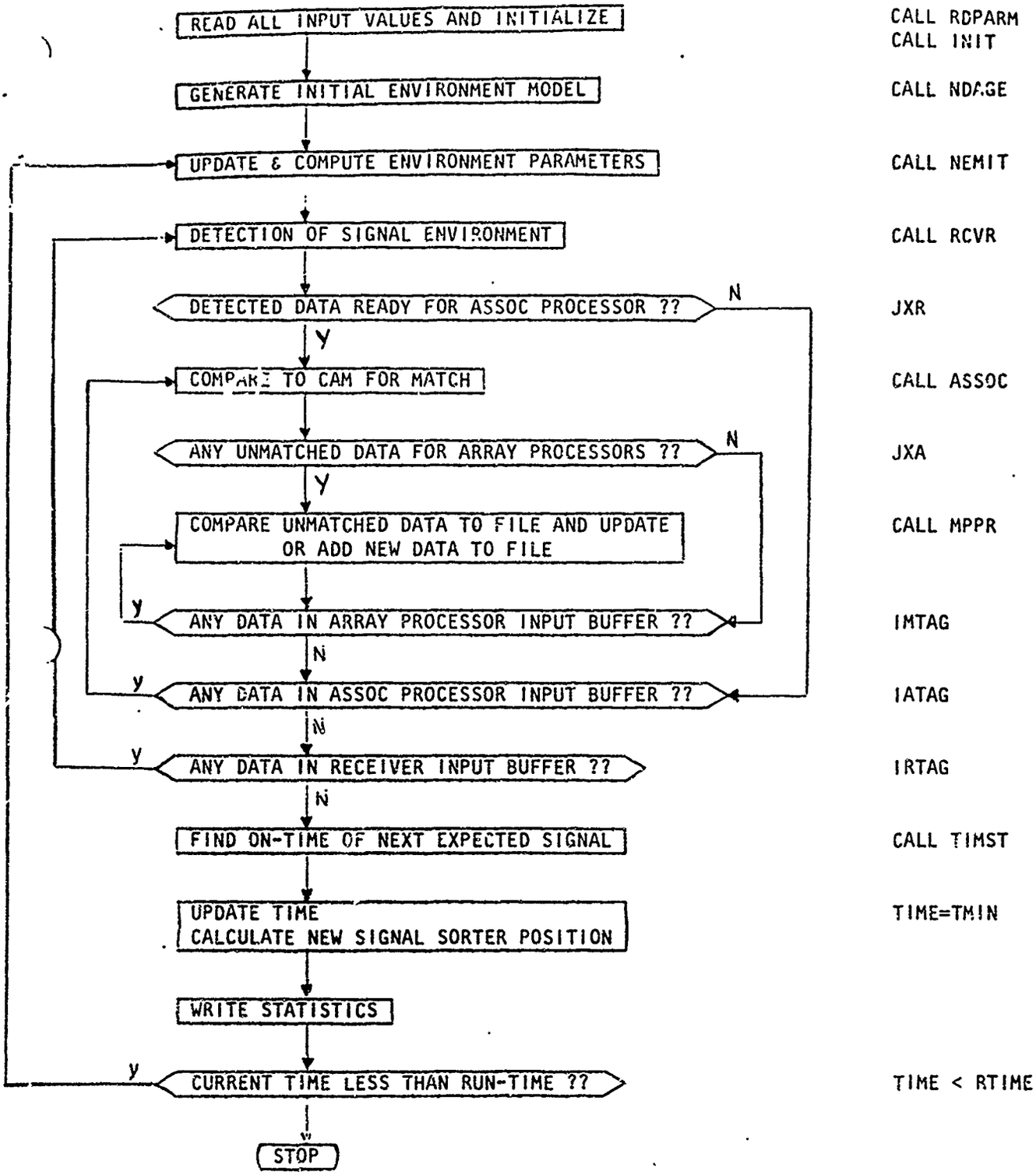
The main program, File EVS4, maintains the calling sequence for the various processors in the system. It consists of 15 subprograms, as outlined below:

RDPARM - reads in system and environmental parameters for each simulation run.
INIT - initializes all variables and arrays.
TIMST - returns the minimum next on-time of an emitter to Main.
NDAGE - establishes initial values for all emitters.
NEMIT - computes and/or updates the parameters measured by the Receiver for each emitter when it turns on.
RCVR - detects incoming signals, measures each parameter, and stores the data for output to the associative processor.
ASSOC - accepts data from RCVR, compares data to the CAM via CAM, updates the CAM via LCAM and sends unmatched data to the array-processors.
CAM - performs an associative comparison of the top of the CAM buffer and the CAM, and counts the number of matches.
LCAM - loads the CAM when the CAM is not busy, and loads the CAM buffer from the array-processors buffer.
MPPR - the array processors, accepts unmatched data from CAM. types emitters and adds new emitters to its memory. Calls routines UPDAAM and UPDM to update unmatched data, calls PRIP to compute PRI, calls CONFIG to configure the LIST.
UPDM - updates memory when unmatched data from CAM matches main memory as a frequency drift but DOA does not change.
UPDAAM - updates memory when unmatched data from CAM matches main memory as a DOA drift.
PRIP - computes the PRI for emitters in dynamic environments.
CONFIG - reconfigures the LIST based on the maximum and minimum PRIs of the emitters currently in the environment.
WSTAT - writes statistics from the simulated run at specifed intervals.

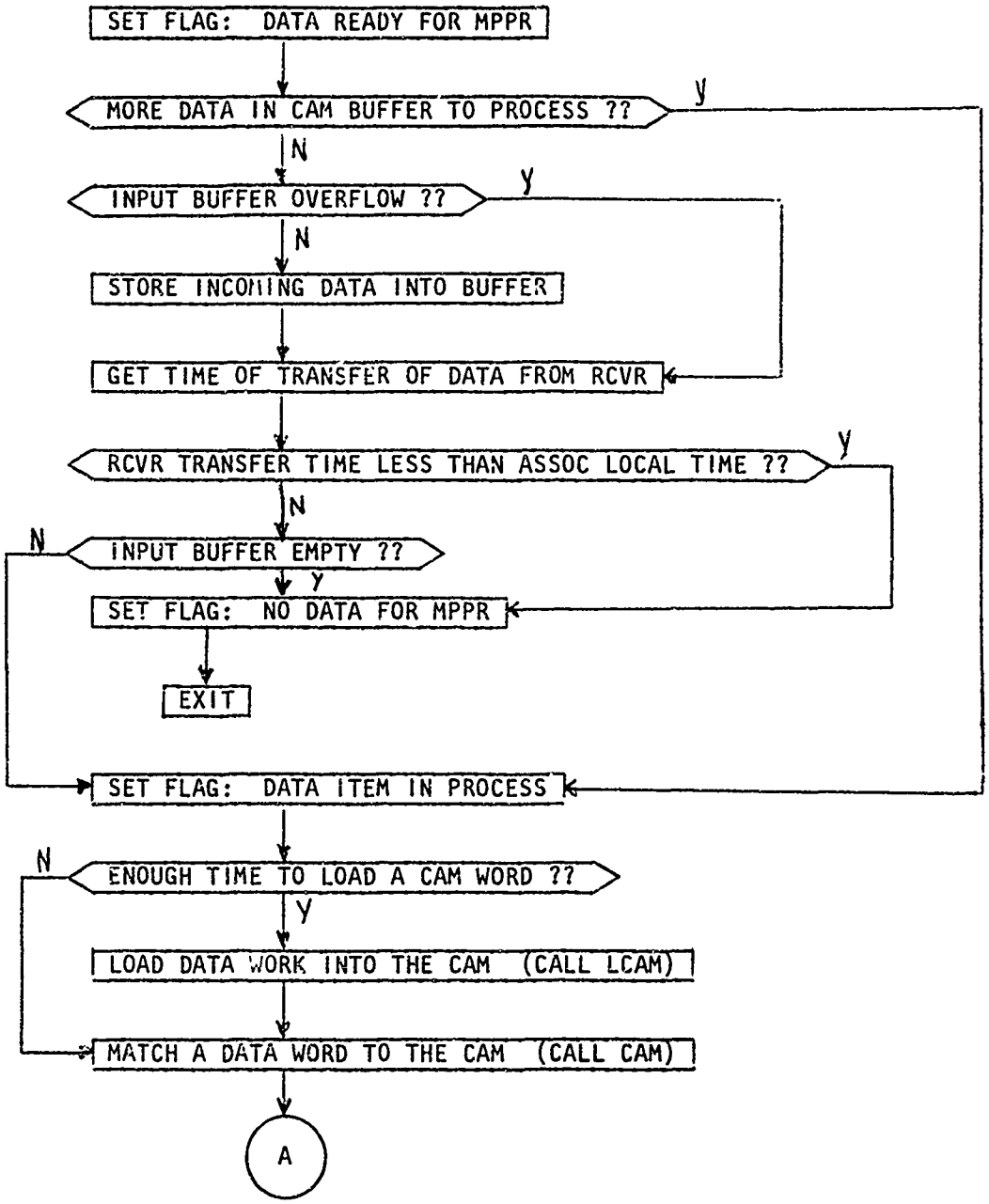
Following are flowcharts for MAIN, ASSOC, LCAM, CAM, MPPR, UPDAAM, and UPDM. Also included is a listing of the entire program.

MAIN ROUTINE

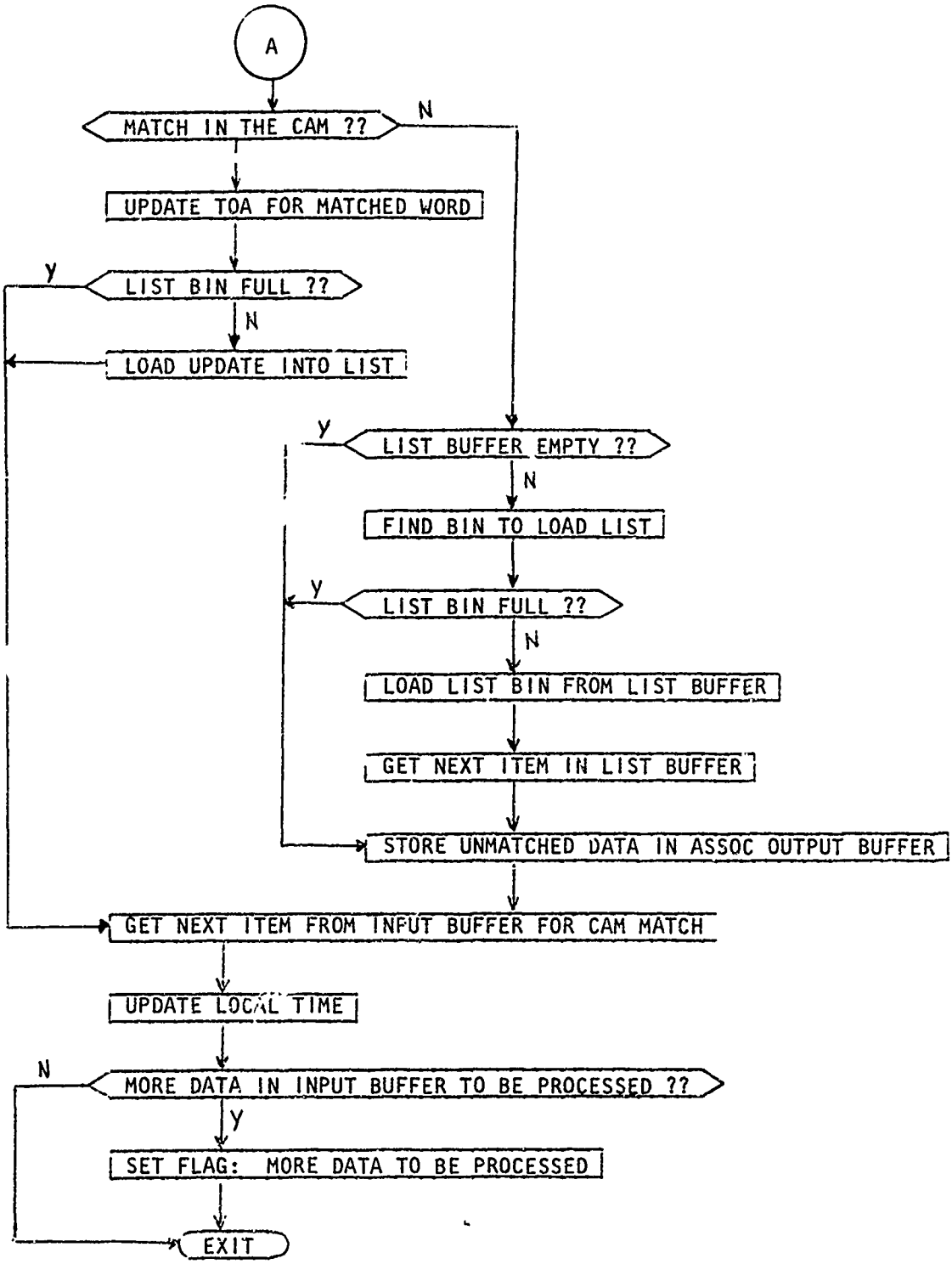
FILE EVS4



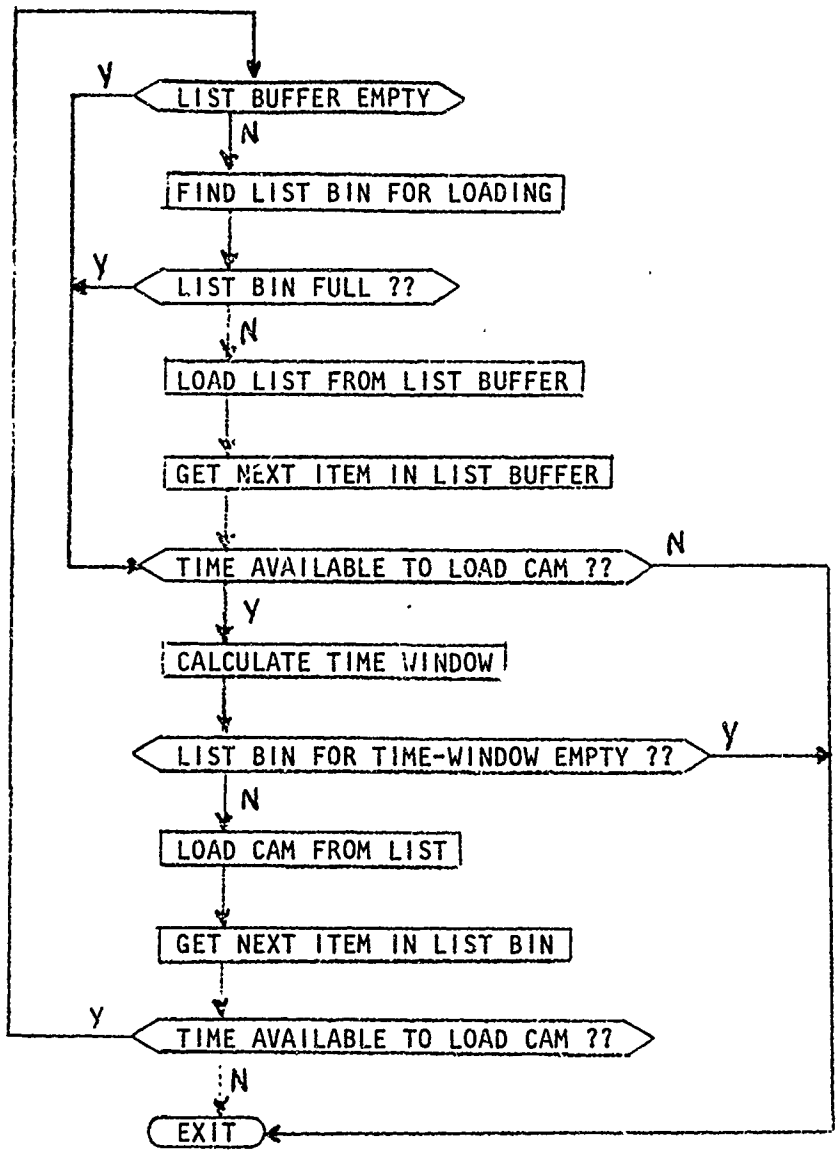
SUBROUTINE ASSOC



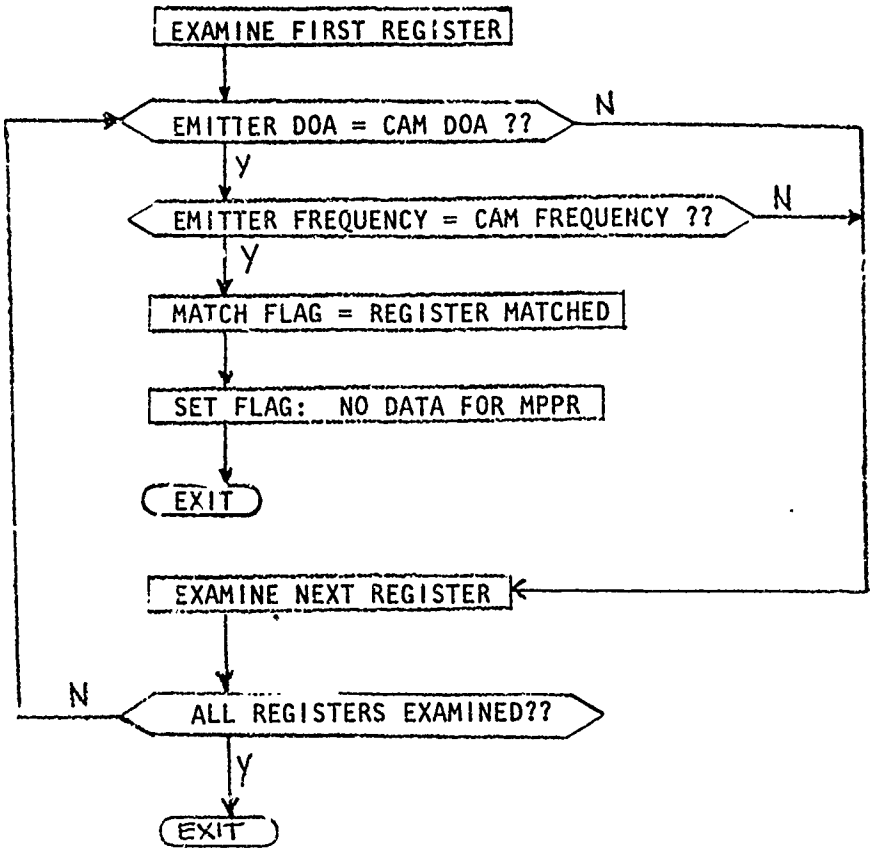
SUBROUTINE ASSOC (CONT.)



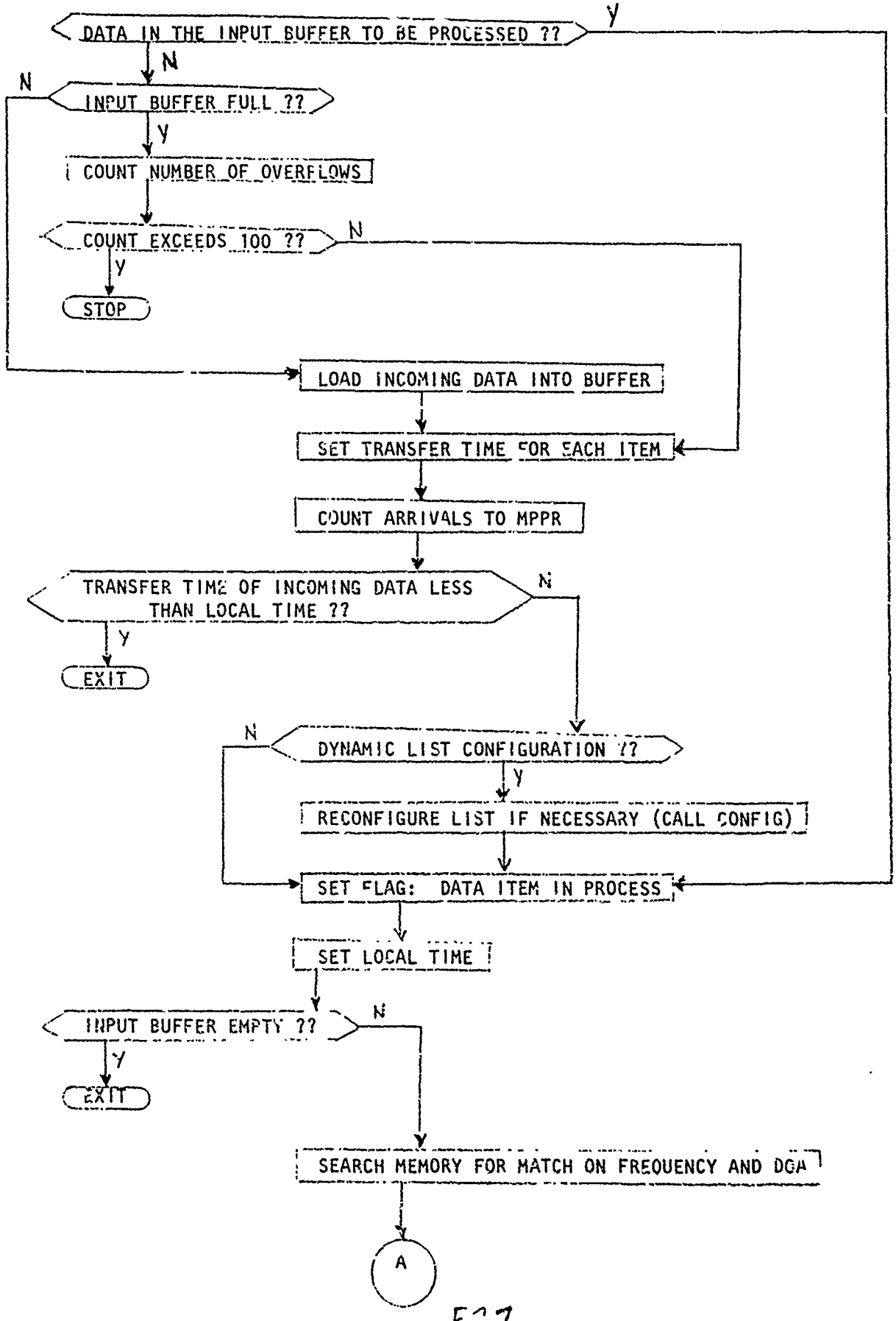
SUBROUTINE LCAM



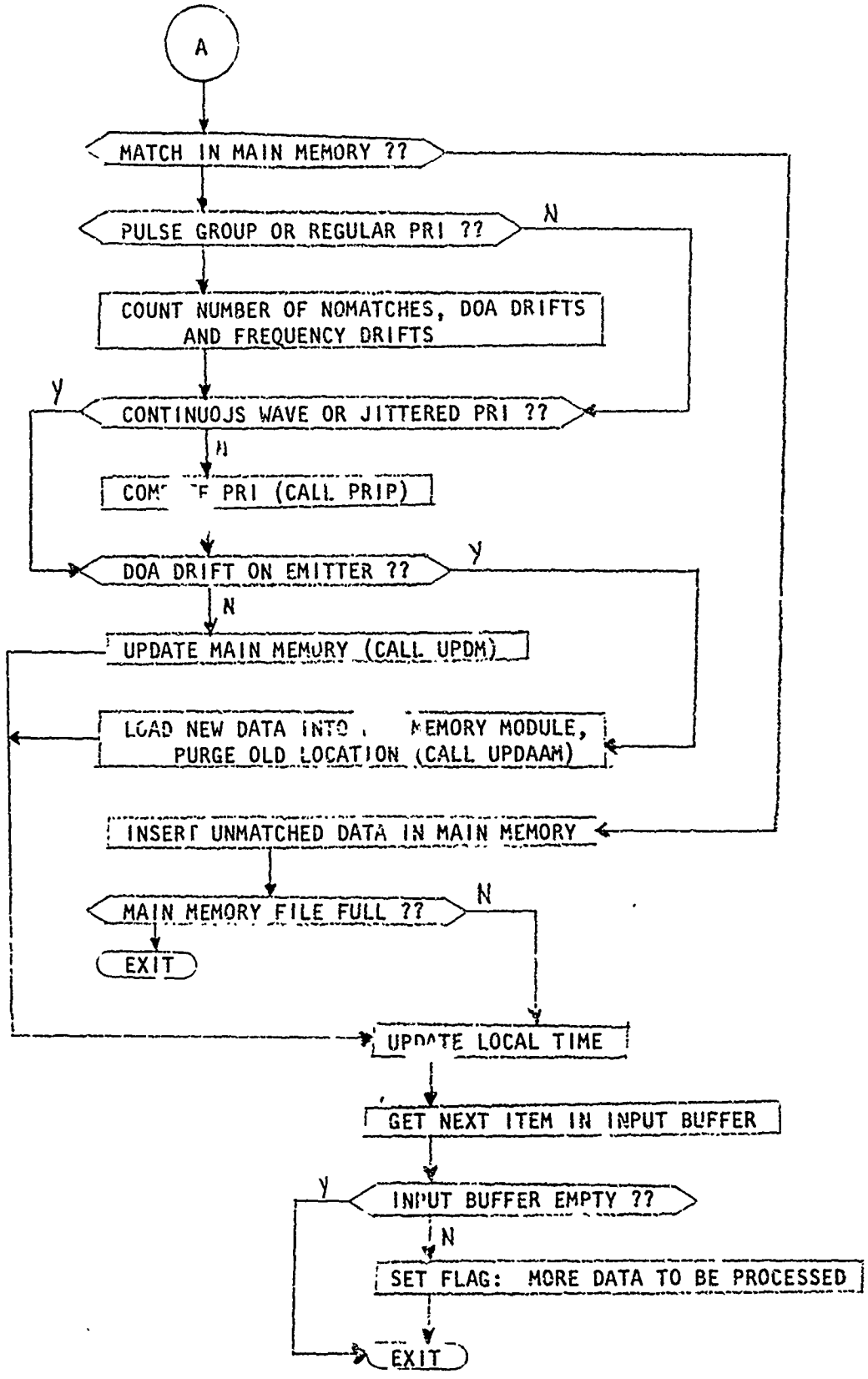
SUBROUTINE CAM



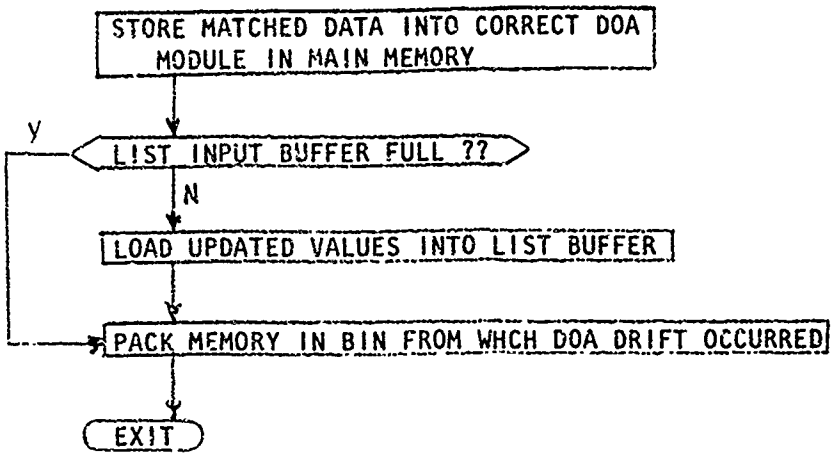
SUBROUTINE MPPR



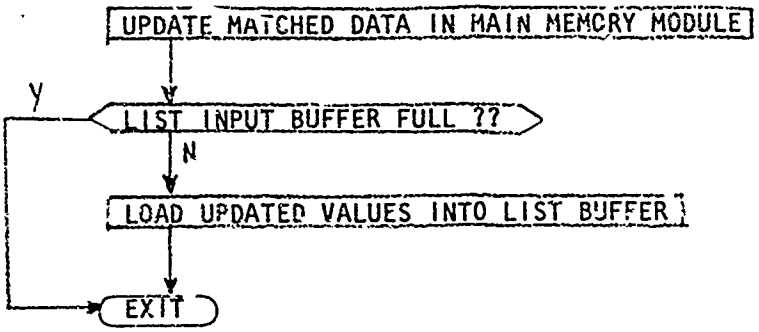
SUBROUTINE MPPR (CONT.)



SUBROUTINE UPDAAM



SUBROUTINE UPDM



```
C     FILE CVS4
C ************
C   THIS PROGRAM [illegible] OR THE LIST
C  ITS CONFIGURATI[illegible] CAMS [illegible] PRIL
C   THIS PROGRAM [illegible] SIMU[illegible] ENVIRONMENT
C ************
C
C
C
C
C     MAIN SUPER[illegible] ROUTINE  FILE [illegible]
C     IMPLICIT INTEGER*4 (I-N)
      REAL MPTL[illegible]
      INTEGER CAM[illegible] CAMFRQ [illegible] CAMPTR [illegible]
      COMMON/EMI[illegible]MITR(500,25),TIME,[illegible],ZE,CONST,NE,MINPTQ,
     1  EMITON(500) [illegible]MIN
      COMMON/RC[illegible]IFREQ[illegible]ITOA,INUM,SL(100),A,B,IRTAG,R[illegible]
      COMMON/AS[illegible]IATAG,A[illegible]XF,IYF(6)[illegible]FIA,MAS,IWRITE,LAD
      COMMON/MP[illegible](64,48,2)[illegible]L,IMUE,IMTAG,IMXF,MAIBUF(64,6),T[illegible]
     1     NWD
     1  ,IRIT,PNT(64),MPNT(64),NIA
      COMMON/UP/[illegible]D(64),CAMFRQ(64),I[illegible]PR(64),CAMPTR
      COMMON/DA/[illegible]
      COMMON/PR[illegible] PRIL,PRIH,NEID,NFCUN
      COMMON/UP[illegible]BUF(32,5)
      COMMON/LAS[illegible]ISF,TLOAD,TIFM,NEA
      COMMON/STAT/[illegible]T,MCNT,MAXQ,MAXCAM,MAXMPP,TMAXTQ,NCWL,TNLM
      COMMON/STAT1/[illegible]DUAD,ICFD
      COMMON/RAND/IRAN,JRAN
      COMMON/LRCVR/[illegible]BUF(64),LRIBUF(64,5) IACELL(16),ACELL(16),
     1    PIB(64,5) [illegible]RIB(64),LRIPT,[illegible] LRISIZ,LRIT
      COMMON/LASFR[illegible]XFR(32),LASSIZ,AS[illegible]LOC,ASTLOP,IDEL,NMOD,
     1  NWPM,NEIT
      COMMON/LMP[illegible]XFR(192),LMISIZ [illegible]LOC,IPCNT
      COMMON/F[illegible]BUFF(64,4),INL
     1    ,LODCAM(256,64,3),LDCAMP(256),LDCAMX(256),
     2    INLT1,INLT2,INLT3
      COMMON/CMAIN/TPRINT,PINC,ETIME,PTIME,VR
      COMMON/JMP[illegible]/JICW,JZAG,JIFH,JIDNA,JZPRI
C
C
C  THIS ROUTINE CALLS THE MAJOR SUBROUTINES, WHICH CALL OTHER
C  SUBROUTINES AS NOTED.  FOR DETAILS, SEE THE PARTICULAR SUBROUTINE
C     RDPARM
C     INIT
C     NEMIT
C     RCVR
C     ASSOC - CALLS CAM & LCAM
C     MPPR  - CALLS CONFIG,PRIP,UPDAAM & UPDM
C     TIMST
C  IT ALSO MAINTAINS THE TIMING SEQUENCE FOR THE PROCESSORS AND
C  PRINTS OUT THE STATISTICAL DATA FOR EACH SIMULATION.
C
C  KEY VARIABLES
C       JXR:    RCVR FLAG: 0-DATA READY FOR ASSOC, 1-NO DATA READY
C       JXA:    ASSOC FLAG. 0-DATA READY FOR MPPR, 1-NO DATA READY
C       IMTAG.  MPPR FLAG: 1-MORE DATA IN BUFFER TO BE PROCESSED
C                          0-A DATA ELEMENT IS BEING PROCESSED
C       IATAG:  ASSOC FLAG. 1-MORE DATA IN BUFFER TO BE PROCESSED
C                          0-A DATA ELEMENT IS BEING PROCESSED
C       IRTAG.  RCVR FLAG   1-MORE DATA IN BUFFER TO BE PROCESSED
C                          0-A DATA ELEMENT ELEMENT IS BEING PROCESSED
C       TIME.   TIME OF ARRIVAL OF SIGNAL - CURRENT
C       [illegible]
```

```
C       TMS:     DIFF  BETWEEN CURRENT AND LAST TOA
C       EMITR(L,8):  PULSE REPETITION INTERVAL OF EMITTER L
C       EMITON(L):   ON-TIME OF EMITTER L
C       XR       X COORDINATE POSITION OF SIGNAL SORTER PLATFORM
C       VR:      VELOCITY OF SIGNAL SORTER PLATFORM IN X DIRECTION
C       TMIN:    PARAMETER TO TEST FOR TIME ABOVE
C
C START
        CALL RDPARM
        CALL INIT
        CALL NDAGE
        DO 30 I=1,NE
30      EMITON(I)=EM(IP(I,14)

100     CALL NEMIT
60      CALL NOVR
        IF (JXR.GE.1.00 TO 40
50      CALL ASSOC
        IF (JXA.GE.1) GO TO 60
70      CALL MPRP
80      IF (IMTAG.NE.1) GO TO 70
40      IF (IATAG.NE.1) GO TO 50
        IF (IRTAG.NE.1. GO TO 60
90      CALL TIMST
        TIME=TMIN
        EMITON(MINPTR)=EMITON(MINPTR)+EMITR(MINPTR,8)
        TMS=TIME-FTIME
        XR=XR+VR*TMS
        FTIME=TIME
        CALL WSTAT
        IF(TIME.LT.RTIME) GO TO 100
        STOP
        END
```

```
C*****RDPARM******************************************************
C*****************************************************************
      SUBROUTINE RDPARM
C     IMPLICIT INTEGER*4 (I-N)
      COMMON/EMIT/EMITR(50,25),TIME,XX,VR,R,CONST,NE,MINPTR,
     1  EMITON(50),TMIN
      COMMON/AS/IW,IATAG,AIXF,IAT,...,NAS,IWRITE,LAD
      COMMON/MP/...M(64,48,8),LL,IMOD,IKTAG,TMXF,MAIBUF(64,...),T,
     1     ...D
     1  ,IT,PNT(64),MPNT(64),TIA
      COMMON/DA/EMS
      COMMON/PRI/...,PRIL,PRIH,NEID,...
      COMMON/LASO/...TSP,TLO...,TIFM,NEA
      COMMON/LASFR/FIXFR(30),LAISIZ,ASILOC,ASILOP,IDEL,NMOD,
     1  NWPM,NBIT
      COMMON/CMAIN/IPRINT,FINC,RTIME,...TIME,VR
      COMMON/JMFZE/JZCW,JZPG,JZFH,JZDOA,JZPRI
C
C
C  THIS SUBROUTINE READS IN THE PARAMETERS FOR EACH SIMULATION RUN
C  KEY VARIABLES
C     IWRITE,IRIT     1-WRITE, 0-SUPPRESS OUTPUT
C     JZCW:           TO GENERATE CONTINUOUS WAVE EMITTERS
C     JZPG:           TO GENERATE PULSE GROUP EMITTERS
C     JZFH:           TO GENERATE FREQUENCY HOPPERS
C     JZDOA           TO GENERATE NONUNIFORM DOA CHANGE
C     JZPRI.          TO SET PRIH AND PRIH MANUALLY
C     NE:             NUMBER OF EMITTERS
C     PRIH:           MAXIMUM PULSE REPETITION INTERVAL
C     PRIL.           MINIMUM PULSE REPETITION INTERVAL
C     EMS:            MAX. SEPARATION BETWEEN INITIAL ON-TIMES
C     TIA :           ASSOC  PROC. TIME IN MICROSECONDS/INSTRUCTION
C     TIM:            MPFR. PROC  TIME IN MICROSECONDS/MICROINSTRUCTION
C     TIFM:           CAM MANAGER PROCESSING TIME IN MICROSECONDS
C     NAS:            NUMBER OF CAM REGISTERS
C     IDEL:           MAXIMUM DELAY BEFORE PROCESSING WORD OF DATA
C     NMOD:           INITIAL NO. OF MODULES IN THE (LODCAM) LCAM STACK
C     NWPM:           NUMBER OF WORDS PER MODULE IN (LODCAM) LCAM STACK
C     NBIT:           INITIAL NUMBER OF BITS SHIFTED
C     LAD:            ADVANCE LOAD TIME IN MICROSECONDS
C     NECON.          NO. OF EMITTERS IDENTIFIED BEFORE CONFIGURATION -
C                     IF < NE, DYNAMIC CONFIG.,  IF >= NE, STATIC
C     RTIME:          RUNTIME FOR SIMULATION IN SECONDS
C
C
C START
      WRITE(6,2)
2     FORMAT(1X,'TYPE 0 FOR DATAFILE INPUT, 1 FOR TERMINAL INPUT')
      READ(1,1)IFILE
      IF(IFILE.GE.1)WRITE(6,21)
21    FORMAT(2X,'TYPE 1 TO WRITE ALL OUTPUT,
     10 FOR ESSENTIAL DATA ONLY')
      READ(1,1)IWRITE
      IRIT=IWRITE
      IF(IFILE.GE.1)WRITE(6,100)
100   FORMAT(1X,'CONTINUOUS WAVE EMITTERS? 1=YES, 0=NO')
      READ(1,1)JZCW
      IF(IFILE.GE.1)WRITE(6,110)
110   FORMAT(1X,'PULSE GROUP EMITTERS?')
      READ(1,1)JZPG
      IF(IFILE.GE.1)WRITE(6,120)
120   FORMAT(1X,'...NONUNIFORM DOA CHANGE?')
      READ(1,1)...
```

```
      IF(IFILE.GE.1)WRITE(6,130)
130   FORMAT(1X,'NUMBER OF FREQUENCY HOPPING EMITTERS?')
      READ(1,1)JF
      IF(IFILE.GE.1)WRITE(6,1)
1     FORMAT(1X,'C=1 HIGH AND LOW PRI  1=YES,0=ACCEPT DEFAULT')
      READ(1,1)JZPRI
      IF(IFILE.GE.1)WRITE(6,2)
      READ(1,1)NE
2     FORMAT(5X,'NUMBER OF UNITORS =
      IF(JZPRI.EQ.1)GO TO 330
      IF(IFILE.GE.1)WRITE(6,333)
333   FORMAT(5X,'HIGEST PRI (IN MILL  SEC.)=')
      READ(1,5)PRIH
      IF(IFILE.GE.1)WRITE(6,444)
444   FORMAT(5X,'LOWEST PRI (IN MILL  SEC.) =')
      READ(1,5)PRIL
330   IF(IFILE.GE.1)WRITE(6,4)
4     FORMAT(5X,'MAX  SEPARATION DET  (IN TIMES=')
      READ(1,5)EMS
5     FORMAT(F8.4)
1     FORMAT(I5)
11    FORMAT(F20.10)
      IF(IFILE.GE.1)WRITE(6,24)
24    FORMAT(1X,'ALLOC  PROC. TIME IN MICROSEC')
      READ (1,11)TIA
C     TIA=1.
      IF(IFILE.GE.1)WRITE(6,14)
14    FORMAT(1X,'MAP PROC  TIME IN MICROSEC PER MICROINSTRUCTION
     *')
      READ(1,11)TIM
C     TIM=0.01
      IF(IFILE.GE.1)WRITE(6,22)
22    FORMAT(1X,'C-M' MANAGER PROC TIME IN MICROSEC')
      READ(1,11)TIFM
C     TIFM=1.
      IF(IFILE.GE.1)WRITE(6,23)
23    FORMAT(1X,'N IAM REG.=')
      READ(1,1)N48
C     N48=24
C     IF(IFILE.GE.1)WRITE(6,31)
31    FORMAT(1X,'DELAY= ')
C     READ(1,1)IDEL
      IDEL=0
      IF(IFILE.GE.1)WRITE(6,32)
32    FORMAT(1X,'INTIAL # OF MOD. IN THE LOAM STACK = ')
      READ(1,1)NMOD
C     IF(IFILE.GE.1)WRITE(6,33)
33    FORMAT(1X,'# OF WORDS/MOD  IN THE LOAM STACK =')
C     READ(1,1)NWPM
      NWPM=64
      IF(IFILE.GE.1)WRITE(6,34)
34    FORMAT(1X,'INITIAL # OF BITS SHIFTED =')
      READ(1,1)NBIT
      IF(IFILE.GE.1)WRITE(6,322)
322   FORMAT(1X,'ADVANCE LOAD TIME IN MICROSEC. =')
      READ(1,1)LAD
C     LAD=0
      IF(IFILE.GE.1)WRITE(6,354)
354   FORMAT(1X,'NUMBER OF EMITTERS IDENTIFIED BEFORE CONFIG =')
      READ(1,1)NEMI
      IF(IFILE.GE.1)WRITE(6,20)
20    FORMAT(1X,'RUN TIME IN SEC. ')
      READ(1,1)  RTIME
      IF(IFILE.GE.1)  WRITE(6,  )
```

```
35 .  FORMAT(1X,'PRINT INCREMENT IN SECONDS=')
      READ(1,11   )TINC
      RETURN
      END
```

```
C*****INIT*****
C*****
      SUBROUTINE INIT
      IMPLICIT INTEGER*4 (I-N)
      INTEGER CAMPTR,CAMFRG,CAMPTR,CAMBUF
      REAL MPT,C,
      COMMON/EMIT/EMITR(50),25  TIME,XP  ,Y,  Z,CONST,NE,MINPTR,
     1  EMITON,500  TMIN
      COMMON/RCV/ICHA,IFREQ,IPW,ITOA,INUM,SL(181),A,B,IRTAG,NTXP  ,  TIP
      COMMON/AE/JAA,IATAG,AZYF,IYF(6),JZA,TIA MAS,IWRITE,LAD
      COMMON/MP MANEM(64,48,8),LL,IMSG  IMIAC,IMXF,MAIBUF(64,6),TIM
     1     NWD
     1  ,IRIT,PNT(64),MPNT(64),NIA
      COMMON/UP/LATCOA(64),CAMFRG(64),ILMPRI(64),CAMPTR
      COMMON/DA/DM,
      COMMON/PRI/ INC,PRIL ,RIM,NEID,NACON
      COMMON/UPP  /BUF(32,5)
      COMMON/LAS  /TSP,TLOAD,TIFM,NEA
      COMMON/STAT/NMONT,MONT,MAXQ,MAXCAN,MAXMPP,TMAXTQ,NCWL,INLM
      COMMON/STAT1/LOCAD,ICFD
      COMMON/RAN/,IRAN,JRAN
      COMMON/LRCVR/ACBUF(64),LRIBUF(64,5),IACELL(16),ACELL(16),,
     1   RIB(64,6),IRIB(64),LRIBT,RCTLOC,LRISIZ,LRIT
      COMMON/LASER/RCXFR(32),LAISIZ,ASTLOC,ASTLOP,IDEL,NMOD,
     1  NWPM,NBIT
      COMMON/LMPXR/XFR(126),LMISIZ,MFTLOC,IBCNT
      COMMON/FM/LCSUFF(64,4),INL
     1   ,LODCAM(256,64,3),LDCAMP(256),LDCAMX(256),
     2   INLT1,INLT2,INLT3
      COMMON/CMAIN/IPRINT,PINC,RTIME,PTIME,VR
      COMMON/JMPZER/JICW,JZFG,JZFH,JZDA,JZPRI
C
C
C  THIS SUBROUTINE INITIALIZES MANY OF THE VARIABLES AND ARRAYS
C  WHICH ARE USED IN OTHER SUBROUTINES.  FOR DETAILS ON VARIABLES
C  SEE THE PARTICULAR SUBROUTINE
C  KEY VARIABLES
C    TIM:  SMALLEST NEXT EVENT TIME-MPP PROCESSOR TIME IN SECONDS/(MSTR
C    TIA.  ASSOC PROCESSOR TIME IN SECONDS -CONVERTED FROM MICROSECS
C    ZR.   Z AXIS POSITION OF SORTER PLATFORM
C    CONST:  USED IN RANGE ATTENUATION EQUATION IN SUBR. NEMIT
C    A,B:  LOWER AND UPPER BOUNDS OF RCVR FREQUENCY
C    SL(I):  RECEIVER ANTENNA PATTERN
C    EMITR(1,14):  TIME OF ARRIVAL OF EMITTER I
C
C
C START
      TIM=TIM*10.**(-6)
      TIR=0.1*10.**(-6)
      INL=0
      DO 555 I=1,128
      LDCAMP(I)=0
555   LDCAMX(I)=0
      DO 556 IA=1,64
      DO 556 IB=1,24
      DO 556 IC=1,8
556   MANEM(IA,IB,IC)=0
      DO 557 J=1,64
      MPNT(J)=0
557   PNT(J)=1
      ASTLOP=0.
      TIME=0.
      IBCNT=0
```

```
      PTIME=0.
      TPRINT='IN'
      XP=0
      YP=0.
      TMAXTL=0
      MFTLOF=0
      ATXF=0
      RTXF=0
      RCTLOC=0
      RCTLOL=0.
      LRIBT=0
      JXR=0
      JXA=0
      IBOAD=0
      ICFD=0
      MCNT=0
      NMCNT=0
      NCWL=0
      INLM=0
      IMTAG=0
      IATAG=0
      IRTAG=0
      IRAN=0
      JRAN=0
      MAX0=0
      MAXCAM=0
      MAXMPP=0
      LMISIZ=0
      LRISIZ=0
      LAISIZ=0
      CONST=-1A
      TIA=TIA*10.**(-6)
      TIFM=TIFM*10.**(-6)
      ZR=3000.
      A=10.**9
      B=10.**11
      VR=500.
      NWD=48
      CAMPTR=1
      IF(JZPRI.NE.0) GO TO 1010
      PRIL=2000.
      PRIH=0.
1010  NEID=0
      NEA=0
      NIA=0
      DO 7 I=1,181
7     SL(I)=-1.0667*FLOAT(I-1)
      RETURN
      END
```

```
C*****TIMST*****
C*****
      SUBROUTINE TIMST
C     IMPLICIT INTEGER*4 (I-N)
      COMMON/EMIT/EMITR(500,25),TIME,XR,YR,ZS,CONST,NE
     1   ,MINPTR,EMITON(500),TMIN,TMS
C
C  THIS SUBROUTINE RETURNS THE MINIMUM NEXT ON-TIME TO MAIN VIA
C  TMIN   MINPTR () USED (IN MAIN) AS THE PTR TO THAT EMITTERS PU
C       EMITON() IS THE ON-TIME ARRAY FOR ALL EMITTERS
C
C START
      MINPTR=1
      TMIN=EMITON(1)
      DO50 J=2,NE
      IF(EMITON(J).GE.TMIN) GO TO 50
      TMIN=EMITON(J)
      MINPTR=J
50    CONTINUE
      RETURN
      END
```

```
C*****NDAGE*****
C*****
      SUBROUTINE MAGE
C     IMPLICIT INTEGER*4 (I-N)
      COMMON/EMIT/EMITR(500,25),TIME,XR,YR,ZR,CONST,NE,MINPTP,
     1  EMITON(500),TMIN
      COMMON/RAND/IRAN,JRAN
      COMMON/CMAIN/IFFINT,IFINC,TINC,PTIME,VR
      COMMON/DA/ZM
      COMMON/JMPVEC/JCON,JPRA,JZEN,JMOVA,JPRI
C
C
C  THIS SUBROUTINE ESTABLISHES INITIAL VALUES FOR ALL EMITTERS.
C  FIELD DEFINITION OF THE PARAMETER ARRAY FOR EACH EMITTER
C       EMITR(I,J) WHERE I=EMITTER NUMBER, J=DEFINED BELOW
C        1.  DX - X DISPLACEMENT OF EMITTER AND SIGNAL SORTER (METERS)
C        2.  DY - Y DISPLACEMENT OF EMITTER AND SIGNAL SORTER (METERS)
C        3.  POWER OF EMITTER (INCLUDES ANTENNA GAIN)
C        4.  MAINLOBE SIZE - OF EMITTER IN DEGREES
C        5.  SIDELOBE LOSS - IN DECIBELS DOWN FROM MAIN BEAM GAIN
C        6.  MAX. ANTENNA ANGLE - UP SCAN LIMIT OF ANTENNA IN DEGREES
C        7.  SCAN RATE - IN HZ
C        8.  PRI - PULSE REPETITION INTERVAL IN SECONDS
C        9.  PULSE WIDTH - IN SECONDS
C       10.  FREQUENCY - IN GIGAHZ
C       11.  ON TIME - TIME WHICH THE EMITTER IS TURNED ON (BEFORE
C                      MAXIMUM ON-TIME ELAPSED)
C       12.  OFF TIME - TIME AT WHICH THE EMITTER IS TURNED OFF
C       13.  RCVR POWER - POWER LEVEL FROM EMITTER I SEEN AT THE
C                         RECEIVER ANTENNA
C       14.  TOA - TIME OF ARRIVAL OF TRANSMITTED PULSE AT RCVR ANTENNA
C       15.  DOA - DIRECTION OF ARRIVAL OF TRANSMITTED PULSE AT RCVR ANT.
C       16.  FLAG - SET FOR DURATION OF PULSE
C       17.  INITIAL ANTENNA ANGLE - IN DEGREES
C       18.  DZ - Z DISPLACEMENT OF EMITTER AND SIGNAL SORTER (METERS)
C       19.  MIN. ANTENNA ANGLE - LOWER SCAN LIMIT IN DEGREES
C       20.  VX - X VELOCITY COMPONENT OF SIGNAL SORTER PLATFORM
C       21.  VY - Y VELOCITY COMPONENT OF SIGNAL SORTER PLATFORM
C       22.  VZ - Z VELOCITY COMPONENT OF SIGNAL SORTER PLATFORM
C       23.  TYPE:  -1:MOVING EMITTER, 0 FIXED, +1:COLLISION COURSE
C       24.  SPARE
C       25.  SPARE
C
C
C START
      DO 15 J=1,NE
      EMITR(J,20)=0.
      EMITR(J,21)=0
      EMITR(J,22)=0
 15   EMITR(J,23)=0.
C
C    INITALIZE VELOCITIES OF MOVING EMITTERS
      DO 25 J=1,NE,10
      EMITR(J,23)=-1.
      EMITR(J,20)=RND(IRAN)*200.-100.
      EMITR(J,21)=RND(IRAN)*200.-100
C
      DO 35 J=2,NE,10
35    EMITR(J,23)=+1.
      DO 100 I=1,NE
      EMITR(I,1)=(RND(IRAN)*20.+2.)*10.**3
      EMITR(I,2)=(RND(IRAN)*40.-22.)*10.**3
      EMITR(I,18)=RND(IRAN)*1000.
```

```
      EMITR(I,3)=RND(IRAN)*50 +30
      EMITR(I,4)=RND(IRAN)* 41+ 09
      EMITR(I,5)=RND(IRAN)* 5 +15
      EMITR(I,17)=ATAN2(EMITR(I,2),EMITR(I,1))
      IF(EMITR(I,17).LT.0) EMITR(I,17)=EMITR(I,17)+6.2832
      EMITR(I,6)=EMITR(I,17)+1.5
      EMITR(I,19)=EMITR(I,17)-1.5
      EMITR(I,7)=RND(IRAN)*1.5+ 5
C
C  GENERATE PRI'S (EMITR(I,8)) BETWEEN 500 AND 7900 MICROSECONDS
      EMITR(I,8)=RND(IRAN)
      IF(EMITR(I,8).LT.0.0)EMITR(I,8)=0
      IF(EMITR(I,8).GT.0.99) EMITR(I,8)=1
      EMITR(I,8)=EMITR(I,8)*0.0074+0.0005
      EMITR(I,9)=RND(IRAN)*.000003+.000001
C
C     ***   MODIFY PULSE WIDTH FOR CW EMITTERS   ***
      IF((JZCW.NE.0).AND.(I.EQ.8))EMITR(I,9)=.0001
C
      EMITR(I,10)=RND(IRAN)*10.**10+2.*10.**9
      EMITR(I,11)=RND(IRAN)*EMS
      IF(NE.LE.100) GO TO 880
      IF(I.GT.100)  EMITR(I,11)=EMITR(I,11)+EMS
      IF(I.GT.200) EMITR(I,11)=EMITR(I,11)+EMS
      IF(I.GT.300) EMITR(I,11)=EMITR(I,11)+EMS
C
880   EMITR(I,12)=RTIME
      EMITR(I,13)=-100.
      EMITR(I,14)=EMITR(I,11)
      EMITR(I,15)=0
      EMITR(I,16)=0.
100   CONTINUE
C
      IF(JZPG.EQ.0) GO TO 1110
C   INITALIZE EMITTERS WITH PULSE GROUPS
      DO 20 I=5,NE,20
      N=I+1
      K=I+2
      DO 21 J=1,20
      EMITR(N,J)=EMITR(I,J)
 21   EMITR(K,J)=EMITR(I,J)
      EMITR(N,11)=EMITR(I,11)+EMITR(I,9)*2.
      EMITR(K,11)=EMITR(N,11)+EMITR(I,9)*2.
      EMITR(N,14)=EMITR(N,11)
      EMITR(K,14)=EMITR(K,11)
20     CONTINUE
C
C  ELIMINATE SIDELOBES FOR TRACKING EMITTERS
1110  DO 30 K=3,NE,20
 30   EMITR(K,5)=0
C
C  CALCULATE THE TOTAL OF ALL PRI'S (TPRIG) AND THE AVERAGE PRI (APRI)
      TPRIG=0
      DO 200 I=1,NE
200   TPRIG=TPRIG+EMITR(I,8)
      APRI=TPRIG/NE
      WRITE(6,111) APRI
111   FORMAT(2X,'APRI=',E12.5)
      RETURN
      END
```

```
C*****NEMIT*************************************************************
C***********************************************************************
      SUBROUTINE NEMIT
C     IMPLICIT INTEGER*4 (I-N)
      COMMON EMIT/EMITR(500,23),TIME,XR,YR,ZR,CONST,NE,MINPTR,
     1  EMITON(500),TMIN
      COMMON/RAND/IRAN,JRAN
      COMMON/JMP/JR,JZCW,JZFG,JLFH,JZDOA,JZHK1
C
C
C  THIS SUBROUTINE COMPUTES AND/OR UPDATES THE PARAMETERS MEASURED
C  BY THE RECEIVER OF EACH EMITTER WHEN IT TURNS ON
C  KEY VARIABLES
C    XR,YR,ZR   X,Y,Z COORDINATES OF SIGNAL SORTER PLATFORM
C    DX,DY,DZ   X,Y,Z DISTANCE BETWEEN EMITTER AND S.S. PLATFORM
C    R.   STRAIGHT LINE DISTANCE BETWEEN EMITTER AND PLATFORM
C    SL   SIDELOBE LOSS SEEN BY SIGNAL SORTER
C    ABE    ANGLE BETWEEN CENTERLINE OF EMITTER AND S.S.
C    NE.  NUMBER OF EMITTERS IN THE ENVIRONMENT
C
C    PARAMETERS FOR EMITR(J,X):        (SEE NDAGE)
C   1-DX, 2-DY, 3-EMITTER POWER, 4-MAINLOBE SIZE, 5-SIDELOBE LOSS,
C   6-MAX. ANT  ANGLE  7-SCAN RATE, 8-PRI, 9-PULSE WIDTH,
C   10-FREQUENCY, 11-ON TIME, 12-OFF TIME, 13-RECEIVER POWER,
C   14-TOA, 15-DOA, 16-FLAG, 17-INIT  ANT  ANGLE, 18-DZ,
C   19-MIN. ANT  ANGLE, 20-VX, 21-VY, 22-VZ, 23-EMITTER TYPE
C
C
C START
      DO 3 J=1,NE
      IF(TIME.GT.EMITR(J,12)) GO TO 3
      IF(EMITR(J,11).GT.TIME) GO TO 3
1     EMITR(J,17)=EMITR(J,17)+EMITR(J,7)*EMITR(J,8)
      IF(EMITR(J,17).GE.EMITR(J,6)) EMITR(J,7)=-EMITR(J,7)
      IF(EMITR(J,17).LE.EMITR(J,19)) EMITR(J,7)=-EMITR(J,7)
      IF(EMITR(J,23).LE.0) GO TO 10
C
C   CALCULATE POSITION FOR COLLISION COURSE EMITTER
      DX=EMITR(J,1)-XR
      DY=EMITR(J,2)-YR
      DZ=EMITR(J,18)-ZR
      R=SQRT(DX*DX+DY*DY+DZ*DZ)
      EMITR(J,20)=600.*DX/R
      EMITR(J,21)=600.*DY/R
      EMITR(J,22)=600.*DZ/R
10    IF (EMITR(J,23).EQ.0) GO TO 11
C   CALCULATE POSITION FOR MOVING EMITTER
      EMITR(J,1)=EMITR(J,1)+EMITR(J,20)*EMITR(J,8)
      EMITR(J,2)=EMITR(J,2)+EMITR(J,21)*EMITR(J,8)
      EMITR(J,18)=EMITR(J,18)+EMITR(J,22)*EMITR(J,8)
C   CALCULATE POSITION FOR FIXED EMITTER
11    DX=EMITR(J,1)-XR
      DY=EMITR(J,2)-YR
      DZ=EMITR(J,18)-ZR
C
C  CALCULATE THE DOA
      IF(DY.NE.0) EMITR(J,15)=ATAN2(DY,DX)
      IF((DX.EQ.0).AND.(DY.EQ.0)) EMITR(J,15)=0
      IF(EMITR(J,15).LT.0)  EMITR(J,15)=EMITR(J,15)+6.2832
C
C   DETECTION OF THE SIDELOBE POWER
      SL=0.
      ABE=EMITR(...
```

DYNAMIC TUNING OF A SIGNAL SORTER IN A DENSE ENVIRONMENT

A.M. ABDALLA
THE GEORGE WASHINGTON UNIVERSITY

February 10, 1982

A final report for
NRL Grant N00014-80-C-0572

```
      IF(ABE .GT. 4.712)ABE=ABE-6.2832
      IF(ABE.LT.0.)ABE=ABE+6.2832
      ABE=ABS(EMITR(J,15)-ABE-.1416
      IF(ABE.LE.EMITR(J,4)) GO TO 4
      SL=EMITR(J,5)
C
C     CALCULATION OF RECEIVER POWER [P(J,10)], TIME OF ARRIVAL [T@(J,...
C     JITTERED PRI [P(J,8)] AND UP TIME UPDATE [T@(J,11)]
4       R=SQRT(DX**2+DY**2+DY*DZ**2+DZ*
        EMITR(J,13)=EMITR(J,9)-CONST-2.*ALOG10(R/1852.)-SL
        EMITR(J,14)=EMITR(J,11)+R/(2.9979*10.**5)
C
C       IF(J.EQ.9) EMITR(J,8)=EMITR(J,8)+RND(IRAN)*300.*10.**-6-RND.(IRAN)
C     1 *150.*10.**-6
C
        EMITR(J,11)=EMITR(J,11)+EMITR(J,8)
        IF(EMITR(J,11).LE.EMITR(J,14)) WRITE(6,20) J
20      FORMAT(2X,'PRI TO SMALL IN NEMI.  J=',I3)
C
C     CALCULATION FOR FREQUENCY HOPPING EMITTERS
        IF(J.LE.JZFH) EMITR(J,10)=EMITR(J,10)*(0.8+0.4*RND(IRAN))
3       CONTINUE
60      RETURN
        END
```

```
C*****RCVR*****
C*****
C    FILE RCVR2 FTN
      SUBROUTINE RCVR
C     IMPLICIT INTEGER*4 (I-N)
      COMMON/EMIT/EMITR(500,25),TIME,AR,VR,ZR,CONST,NE,MINPTR,
     1  CMITOR/500,TMIN
      COMMON/RAND/ISAN,JRAN
      COMMON/RCV/IDNA,IFREQ,IPW,ITOA,INUM,SL(191),A,B,IRTAG,RTAF,JA,IP
      COMMON/LRCVR/RCBUF(64),LRIBUF(64,5),IACELL(16),ACELL(16),
     1  RIB(64,5),IRIB(64),LRIBT,RCBLOC,LRIBIZ,LRIT
      COMMON/JMPZER/JZCW,JZPR,JZFR,JZDDA,JZPR1
C
C
C  THIS ROUTINE DETECTS INCOMING SIGNALS, MEASURES EACH PARAMETER
C  AND STORES THE DATA IN THE RECEIVER OUTPUT BUFFER (LRIBUF)
C  KEY VARIABLES
C    JXR:        DATA TRANSFER FLAG. 0-DATA READY FOR ASSOC FROM RCVR
C                                    1-NO DATA READY
C    IRTAG:      PROCESSING FLAG: 1-DATA IN INBUF WAITING TO BE PROCESSED
C                                 0-A BUFFER ELEMENT IS BEING PROCESSED
C    TIME:       SIGNAL ARRIVAL TIME AT RCVR
C    EMITR(J,14):  TOA
C    LRIBT:       CURRENT SIZE OF RCVR INPUT BUFFER (RIB(*,*))
C    RIB(LRIBT,#):  RECEIVER INPUT BUFFER
C                  #1-TOA, #2-PW, #3-DOA, #4-RCVR POWER, #5-FREQ
C    IRIB(LRIBT):  EMITTER ID NUMBER
C    IP:           NUMBER OF EMITTERS ANALYZED IN EACH PASS THROUGH
C                  RECEIVER  (LOCAL VARIABLE)
C    IRF:          LOCAL FLAG. 0-EMITTER IN FIRST BUFFER LOCATION NOT
C                              SEEN, 1-EMITTER SEEN
C    IGF:          LOCAL FLAG. 0-CURRENT EMITTER IDENTIFIED BY IP NOT
C                              SEEN, 1-EMITTER SEEN
C    LRIT:         SORT FLAG, 1-SWITCH MADE, 0-NO SWITCH
C
C
C START
      IF (IRTAG.GE.1) GO TO 40
      JXR=0
C   TEST INPUT STREAM FOR OBSERVABLE DATA
      DO 80 J=1,NE
      IF(EMITR(J,14).GT.TIME)GO TO 80
C   ADD PULSE TO BUFFER AND CLEAR OLD TOA
      IF(LRIBT.LT.64)GO TO 81
      WRITE(6,82)
82    FORMAT(2X,'RCVR IN OVERFLOW')
      GO TO 80
81    LRIBT=LRIBT+1
      RIB(LRIBT,1)=EMITR(J,14)
      RIB(LRIBT,2)=EMITR(J,9)
      RIB(LRIBT,3)=EMITR(J,15)
      RIB(LRIBT,4)=EMITR(J,13)
      RIB(LRIBT,5)=EMITR(J,10)
      IRIB(LRIBT)=J
      EMITR(J,14)=EMITR(J,14)+1.
80    CONTINUE
C   ORDER BUFFER BY TOA IN ASCENDING ORDER
      IF(LRIBT.LE.1) GO TO 69
91    LRIT=0
      KKK=LRIBT-1
      DO 90 JJ=1,KKK
       I=LRIBT-JJ
       J=I+1
```

```
      IF(RIB(I,1) LT RIB(J,1))GO TO 90
      K=IRIB(I)
      IRIB(I)=IRIB(J)
      IRIB(J)=K
      DO 92 K=1,5
      X=RIB(I,K)
      RIB(I,K)=RIB(J,K)
92    RIB(J,K)=X
      LRIT=1
90    CONTINUE
      IF(LRIT GT.0) GO TO 91
C   TEST BUFFER FOR POTENTIAL OVERLAP
93    ALIM=RIB(1,3)+RIB(1,2)
      IF(ALIM.GE.RIB(LRIBT,1))GO TO 70
C   INITALIZE
      DO 50 K=1,16
      IACELL(K)=0
      ACELL(K)=-700.
50    CONTINUE
      IP=1
      KT=101
C   TEST FOR FREQ. BAND
91    IF(RIB(IP,5).GE.A AND RIB(IP,5).LE.B)GO TO 20
      IF(IP.LE.1) GO TO 16
      GO TO 17
C  TEST RECEIVER SENSITIVITY
20    IF(RIB(IP,4).GE.-500.)GO TO 42
      IF(IP.LE.1)GO TO 16
      GO TO 17
C   COMPUTE THE POWER IN EACH ICELL
42    ANGLE=1.5708
      DO 2 ICELL=1,16
      ANGLE=ANGLE-.1963
      IF(ANGLE.LT.0 )ANGLE=6.2832+ANGLE
      THETA=ABS(RIB(IP,3)-ANGLE)
      IF(THETA.GT.3.1416) THETA=6.2832-THETA
      ITHETA=THETA*57.2958+1.5
      PT=RIB(IP,4)+SL(ITHETA)
C   TEST POWER AGAINST PRIOR LEVEL
      IF(PT.LT.ACELL(ICELL)) GO TO 2
      IACELL(ICELL)=IRIB(IP)
      ACELL(ICELL)=PT
2     CONTINUE
      IRF=0
      IOF=0
      PT=-500.
C   FIND BEAM WITH MAX POWER
1     DO 10 K=1,16
      IF(IACELL(K).EQ.101) GO TO 10
      IF(IACELL(K).EQ.IRIB(IP)) IOF=1
      IF(IACELL(K).NE.IRIB(I)) GO TO 10
      IRF=1
      IF(IP.NE.1) GO TO 10
      IF(ACELL(K).LT.PT) GO TO 10
      KT=K
      PT=ACELL(K)
10    CONTINUE
C   INTERPOLATE NO.
      IF(IP.NE.1) GO TO 33
      LU=KT+1
      LL=KT-1
      IF(KT.EQ.1) LL=16
```

```
      IF(ACELL(LL) GE.ACELL(LU))GO TO 31
32    X=ACELL(KT)-ACELL(LU)
      IF(X GT.11 9) IDEL=0
      IF(X LE.11 9) IDEL=1
      IF(X.LE.6 04) IDEL=2
      GO TO 33
31    X=ACELL(KT)-ACELL(LL)
      IF(X GT.11 9) IDEL=1
      IF(X.LE.11 9) IDEL=0
      IF(X.LE.6 02) IDEL=-1
33    CONTINUE
C   TEST FOR PULSE STILL PRESENT
      IF(IRF.GT.0) GO TO 14
      IF(IP LE.1) GO TO 16
C   MODIFY PULSE WIDTH & ADD TO BUFFER
      RIB(1,2)=RIB(IP,1)-RIB(1,1)
      GO TO 18
14    IF(IP.LE.1) GO TO 17
      IF(IOF.GT.0) GO TO 17
C   MODIFY SUBSEQUENT ARRIVALS
      RIB(IP,1)=ALIM
17    IP=IP+1
C   TEST FOR OVERLAP
      IF(ALIM.GT.RIB(IP,1))GO TO 51
C   TEST BUFFER OVERFLOW
18    IF(LRISIZ.LT 64) GO TO 41
      WRITE(6,83)
83    FORMAT(2X,'DRCVR OUT OVERFLOW')
      GO TO 16
C/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
C
C KEY VARIABLES:
C   LRISIZ:        CURRENT SIZE OF OUTPUT BUFFER (LRIBUF)
C   LRIBUF(*,#)    #1-DOA, #2-FREQ, #3-PW, #4-TOA, #5-EMITTER ID#
C   IDOA, IFREQ, IPW, ITOA, INUM:
C                  OUTPUT VARIABLES FROM RCVR.  INUM=EMITTER ID#
C   RCBUF(*):      TIME AT WHICH PROCESSING OF DATA WORD * IS COMPLETED
C   RCTLOC:        LOCAL TIME IN RCVR
C   RTXF:          RCTLOC FOR TRANSFER TO ASSOC
C   TIR:           PROCESS STEP TIME OF RCVR (.1E-6)
C
C/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
41    LRISIZ=LRISIZ+1
C   QUANTIZE THE DOA BETWEEN 1 AND 64
      LRIBUF(LRISIZ,1)=(KT*4)-2+IDEL
      IF(JZDOA.EQ.0) GO TO 52
C   ***** MODIFY DOA ***** ONLY FOR THOSE EMITTERS WITH DOA=8
      IF(LRIBUF(LRISIZ,1) NE.8)  GO TO 52
      Y=-RND(IRAN)*3.+1.
      IF(Y.LT.-1) Y=-1.
      IF(Y.GT.1) Y=1.
      IDOACK=Y
      LRIBUF(LRISIZ,1)=LRIBUF(LRISIZ,1)+IDOACK
C  ****    ****     ****     ****     ****     ****     ****
C   QUANTIZE THE FREQ BETWEEN 1 AND 4096
52    X=RIB(1,5)-2.*10.**9
      LRIBUF(LRISIZ,2)=X*4095./(10.**10)+1
C   QUANTIZE THE PULSE WIDTH
      LRIBUF(LRISIZ,3)=RIB(1,2)*.5*10.**6
C   QUANTIZE THE TOA
      TOAD=RIB(1,1)*10.**6
67    IF(TOAD.LT.2.**15) GO TO 68
      TOAD=TOAD-2.**15
```

```
      GO TO 67
C
68    CONTINUE
      LRIBUF(LRISIZ,1)=TDAD
      RCBUF(LRISIZ)=RIB(1,1)+RIB(1,2)
C   STORE EMITTER NUMBER
      LRIBUF(LRISIZ,5)=IRIB(1)
C   POP UP STACK
      DO 19 J=2,LRIBT
      I=J-1
      IRIB(I)=IRIB(J)
      DO 19 K=1,5
      RIB(I,K)=RIB(J,K)
19    CONTINUE
      LRIBT=LRIBT-1
      GO TO 93
C   TEST FOR MODULE PROCESSING COMPLETE
70    IF (RCBUF(LRISIZ).LT.RCTLOC) GO TO 69
      IF(RCTLOC.NE.0.) RCTLOC=RCBUF(1)+TIR
      IF(LRISIZ.GT.1) GO TO 40
      RCTLOC=RCBUF(1)+TIR
69    JXR=1
      GO TO 15
40    IRTAG=0
C   STORE PROCESSED ELEMENTS IN OUTPUT ARRAY
      IDOA=LRIBUF(1,1)
      IFREQ=LRIBUF(1,2)
      IPW=LRIBUF(1,3)
      ITOA=LRIBUF(1,4)
      INUM=LRIBUF(1,5)
      RTXF=RCTLOC
C   POP STACK UP
      LRISIZ=LRISIZ-1
      DO 30 I=1,LRISIZ
      K=I+1
      RCBUF(I)=RCBUF(K)
      DO 30 J=1,5
      LRIBUF(I,J)=LRIBUF(K,J)
30    CONTINUE
C   RCTLOC=TIME WHEN PROCESSING COMPLETED
      X=RCBUF(1)-RCTLOC
      IF (X.LT.0.) RCBUF(1)=RCTLOC
      RCTLOC=RCBUF(1)+TIR
C   TEST FOR MODULE PROCESSING COMPLETE
      T=RCBUF(LRISIZ)-RCTLOC
      IF(T.GE.0 .AND.LRISIZ.GE.2) IRTAG=1
15    RETURN
      END
```

```
C*****ASSOC*****
C*****
C    FILE ASPR2.FTN
      SUBROUTINE ASSOC
C     IMPLICIT INTEGER*4 (I-N)
      INTEGER CAMDOA,CAMFRQ,CAMPTR,CAMBUF
      COMMON/RCV/IDOA,IFREQ,IPW,ITOA,INUM,SL(101),A,B,IRTAG,RTXF,JXR,TI
      COMMON/AS/NAS,IATAG,ATXF,IXF(6),JXA,TIA,MAS,IWRITE,LAD
      COMMON/UP/CAMDOA(64),CAMFRQ(64),ICMPRT(64),CAMPTR
      COMMON/UPS/CAMBUF(32,5)
      COMMON/LASOR/TSF,TLOAD,TIFM,NEA
      COMMON/FM/LCBUFF(64,4),INL,
     1   LODCAM(20,64,3),LDCAMP(256),LDCAMX(256),
     2  INLT1,INLT2,INLT3
      COMMON/STAT/NMCNT,MCNT,MAXQ,MAXCAM,MAXMPP,TMAXTQ,NCWL,INLM
      COMMON/LASPR/RCXFR(32),LAISIZ,ASTLOC,ASTLOP,IDEL,NMOD,NWPM,NST
C
C
C ASSOC ACCEPTS DATA FROM RCVR, PERFORMS AN ASSOCIATIVE COMPARISON
C AND SENDS UNMATCHED DATA TO MPPR.
C KEY VARIABLES:
C    JXA:              DATA TRANSFER FLAG:  0-DATA READY FOR MPPR FROM ASSOC
C                                           1-NO DATA READY
C    IATAG:          PROCESSING FLAG:  1-MORE DATA IN CAMBUF TO BE PROCESSED
C                                      0-A BUFFER ELEMENT IS BEING PROCESSED
C                          NEW DATA
C    LAISIZ:           POINTER TO SIZE OF ASSOC INPUT BUFFER (CAMBUF)
C    CAMBUF(LAISIZ,#).
C                      #1-DOA, #2-FREQ, #3-PW, #4-TOA, #5-EMITTER ID#
C    NEA:              NUMBER OF EXTERNAL ARRIVALS
C    RCXFR(LAISIZ):    TRANSFER TIME FOR EACH ELEMENT FROM RCVR
C    ASTLOC:           LOCAL TIME FOR ASSOC
C    TIFM:             INSTR TIME TO LOAD A DATA WORD INTO THE CAM
C    ASTLOP:           PREVIOUS EXIT TIME FROM ASSOC
C    TLOAD:            TIME AVAILABLE TO LOAD DATA INTO THE CAM
C    MAS:              CAM MATCH FLAG:  0-NO MATCH, 1-MATCH
C    MCNT:             NUMBER OF WORDS MATCHED
C    LCSM:             (LOCAL VAR) POINTER TO MODULE NUMBER FOR LODCAM
C    LDCAMP(LCSM)       POINTER FOR MODULE LCSM
C    LCBUFF(1,#)        LOAD BUFFER TO THE LIST
C                      #1-DOA, #2-FREQ, #3-NTOA, #4-PRI
C    LODCAM(*,*,#):     (LIST)  #1-DOA, #2-FREQ, #3-PRI
C    TIA.              INSTRUCTION TIME TO COMPARE A WORD WITH THE CAM
C    IXF(1-6):         OUTPUT ARRAY FOR UNMATCHED WORDS
C                      #1-DOA, #2-FREQ, #3-PW, #4-TOA, #5-EMITTER ID#,
C                           #6-MATCH FLAG (MAS)
C    INL:              POINTER TO SIZE OF LCBUFF STACK
C    ATXF:             TRANSFER TIME OF DATA FROM ASSOC TO MPPR
C
C
C START
      JXA=0
C     IF (JXR.EQ.0) GOTO 7
      IF (IATAG.GE.1) GO TO 20
C   CHECK FOR BUFFER OVERFLOW
7     IF(LAISIZ.LT.32) GO TO 11
      IF(IWRITE  GT.0) WRITE(6,12)
12    FORMAT(2X,'ASPR OVF')
      GO TO 21
C   STORE INCOMING DATA WORDS IN BUFFER
11    LAISIZ=LAISIZ+1
      CAMBUF(LAISIZ,1)=IDOA
      CAMBUF(LAISIZ,2)=IFREQ
```

```
      CAMBUF(LAISIZ,3)=IPW
      CAMBUF(LAISIZ,4)=ITOA
      CAMBUF(LAISIZ,5)=INUM
      NEA=NEA+1
      RCXFR(LAISIZ)=RTXF
C   TEST FOR MAX BUFFER SIZE
      IF(LAISIZ.GT.MAXCAM) MAXCAM=LAISIZ+1
C   TEST FOR MODULE PROCESSING COMPLETE
      IF (RCXFR(LAISIZ).LT.ASTLOC) GO TO 19
      IF(ASTLOC.LT.RCXFR(1)) ASTLOC=RCXFR(1)+TIA
      IF(LAISIZ.GT.1) GO TO 20
19    JXA=1
C
      RETURN
C
20    IATAU=0
C   TEST FOR CAM LOADING
      TLOAD=RCXFR(1)-ASTLOC
      IF(TLOAD.LE.TIFM) GO TO 55
      CALL LCAM
C  TEST FOR ASSOCIATIVE MATCH
55    CALL CAM
C   CHECK FOR WORD MATCHED
      IF(MAS.LT.1) GO TO 60
      MCNT=MCNT+1
C  LOAD  LODCAM FROM CAMBUF AND UPDATE NTOA
      LCSM=(ICMPRI(MAS)+CAMBUF(1,4))/NBIT
      LCSM=MOD(LCSM,NMOD)+1
      IF(LDCAMP(LCSM).GE.NMPM) GO TO 10
      LDCAMP(LCSM)=LDCAMP(LCSM)+1
      LDUM=LDCAMP(LCSM)
      DO 58 I=1,2
      LODCAM(LCSM,LDUM,I)=CAMBUF(1,I)
58    CONTINUE
      LODCAM(LCSM,LDUM,3)=ICMPRI(MAS)
      TLOAD=TLOAD-TIFM
      GO TO 10
60    IF(INL.LT.1) GO TO 61
      LCSM=LCBUFF(1,3)/NBIT
      LCSM=MOD(LCSM,NMOD)+1
59    IF(LDCAMP(LCSM).GE.NMPM) GO TO 61
      LDCAMP(LCSM)=LDCAMP(LCSM)+1
      LDUM=LDCAMP(LCSM)
      DO 62 I=1,2
62    LODCAM(LCSM,LDUM,I)=LCBUFF(1,I)
      LODCAM(LCSM,LDUM,3)=LCBUFF(1,4)
      TLOAD=TLOAD-TIFM
C  POP LCBUFF STACK
      INL=INL-1
      DO 63 I=1,INL
      DO 63 J=1,4
      II=I+1
63    LCBUFF(I,J)=LCBUFF(II,J)
C  STORE UNMATCHED DATA IN OUTPUT ARRAY
61    DO 65 I=1,5
65    IXF(I)= CAMBUF(1,I)
      IXF(6)=MAS
C   SET TRANSFER TIME
      ATXF=ASTLOC
C   POP STACK UP
10    LAISIZ=LAISIZ-1
      IF(IWRITE.GT.0)WRITE(6,8)(CAMBUF(I,I),I=1,4),LAISIZ,ASTLOC,
     1   CAMBUF
```

```
      .
5     FORMAT(1X,'AOFR',5I10,3X,E10.3,I4)
      DO 70 I=1,LAISIZ
      K=I+1
      ROXFR(I)=ROX-V(K)
      DO 70 J=1,5
      CAMBUF(I,J,MCAMEUF(K,J)
70    CONTINUE
C   UPDATE ASTLOC.
      IF(ROXFR(1).LT.ASTLOC) ROXFR(1)=ASTLOC
C   STORE PREVIOUS LOCAL COMPLETION TIME
      ASTLOP=ASTLOC
      ASTLOC=ROXFR(1)+TIA
C   TEST FOR MODULE PROCESSING COMPLETE
      IF(ROXFR(LAISIZ).GE.ASTLOC AND LAISIZ.GE.2) IATAG=1
      IF(LAISIZ.GE.2) IATAG=1
80    RETURN
      END
```

```
C*****CAM*****
C**********
      SUBROUTINE CAM
C     IMPLICIT INTEGER*4 (I-N)
      INTEGER CAMDOA,CAMFRQ,CAMPTR,CAMBUF
      COMMON/AS/NAS,IATAG,ATXF,IXF(6),JXA,TIA,MAS,IWRITE,LAD
      COMMON/UP/CAMDOA(64),CAMFRQ(64),ICMFR(64),CAMPTR
      COMMON/UP3/CAMBUF(32,5)
C
C  THIS ROUTINE PERFORMS AN ASSOCIATIVE COMPARISON OF THE TOP OF
C  THE CAM BUFFER AND THE CAM.
C  KEY VARIABLES
C    MAS:          MATCH FLAG  0-NO MATCH  1-MATCHWORD PTR IN CAM
C    NAS:          SIZE OF THE CAM (# OF CAM REGISTERS)
C    CAMDOA(I)      DOA FIELD OF WORD I IN CAM
C    CAMFRQ(I):    CARRIER FREQUENCY FIELD OF WORD I IN CAM
C    JXA:          DATA TRANSFER FLAG  0-DATA READY FOR MPPR FROM ASSOC
C                                      1-NO DATA READY
C    CAMBUF(1,1):  DOA FIELD OF CAM BUFFER
C    CAMBUF(1,2)    FREQUENCY FIELD OF CAM BUFFER
C                    CAMBUF(1,3):PW;  CAMBUF(1,4):TOA;  CAMBUF(1,5):ID#
C
C
C START
      MAS=0
55    DO 100 I=1,NAS
      IF(CAMBUF(1,1).NE.CAMDOA(I)) GO TO 100
      IF(CAMBUF(1,2).NE.CAMFRQ(I)) GO TO 100
      IF(IWRITE.GT.0)WRITE(6,11)
11    FORMAT(5X'ASSOC MATCH',I2)
      MAS=I
      JXA=1
      GO TO 80
100   CONTINUE
      IF(IWRITE.GT.0)WRITE(6,5)
5     FORMAT(5X'NO MATCH IN ASSOC')
80    RETURN
      END
```

```
C*****LCAM*****
C**********
C     FILE LCAM3 FTN
      SUBROUTINE LCAM
C     IMPLICIT INTEGER*4 (I-N)
      INTEGER CAMDOA,CAMFRQ,CAMPTR
      COMMON/LASMR/RCVFR(32),LASIZ,ASTLOC,ASTLOP,IDEL,NMOD,NWPM,IJJ
      COMMON/AC/NAC,IATAC,ATXF,IXF(6),JXA,TIM,MAS,IWRITE,LAD
      COMMON/FM/LCBUFF(64,4),INL,
     1    LODCAM(256,64,3),LDCAMP(256),LDCAMX(256),
     2    INLT1,INLT2,INLT3
      COMMON/LASOC/TSF,TLOAD,TIFM,NEA
      COMMON/UP/CAMDOA(64),CAMFRQ(64),ICMPRI(64),CAMPTR
      COMMON/STAT/NMCNT,MCNT,MAXQ,MAXCAM,MAXMPP,TMAXTQ,NCWL,INLM
C
C
C   *THIS ROUTINE LOADS LODCAM FROM LCBUFF AND ALSO LOADS THE CAM
C    FROM LODCAM.
C   *THE CAM IS LOADED ONLY WHEN THE CAM IS NOT DOING A SEARCH.
C    THE CAM SEARCH IS DONE ON REAL-TIME AND HENCE HAS HIGHER
C    PRIORITY THAN THE CAM LOAD.
C   *TIME BETWEEN SEARCHES IS USED TO LOAD THE CAM FROM LODCAM.
C
C   KEY VARIABLES:
C   INL:            POINTER TO SIZE OF LCBUFF
C   NBIT:           NUMBER OF BITS SHIFTED
C   NMOD:           NUMBER OF MODULES
C   NWPM:           NUMBER OF WORDS PER MODULE
C   LCSM:           MODULE NUMBER          (LOCAL TO LCAM)
C   LDCAMP(IT):     POINTER FOR MODULE IT
C   LDCAMX(IT):     MAXIMUM VALUE OF LDCAMP(IT)
C   LODCAM(MOD#,PTR,DATA-ELEMENT).
C                   LIST FOR LOADING CAM.  LOADED FROM LIST BUFFER, LCBUFF
C                   LODCAM(*,*,1):    DOA
C                   LODCAM(*,*,2):    FREQUENCY
C                   LODCAM(*,*,3):    PRI
C   LCBUFF(*,#):    INPUT BUFFER TO THE LIST, INCLUDES DATA MATCHED IN THE
C                   CAM AND  THE ARRAY PROCESSORS
C                   #1-DOA;   #2-FREQ;   #3-NTOA;   #4-PRI
C   TIFM:           INSTRUCTION TIME TO LOAD A DATA WORD INTO THE CAM
C   TLOAD:          TIME AVAILABLE TO LOAD THE CAM
C   CAMPTR:         POINTER TO NEXT CAM-LOAD LOCATION:  CAMPTR-1=CAMSIZE
C   NCWL:           TOTAL NUMBER OF WORDS LOADED INTO THE CAM
C   CAMDOA(*),CAMFRQ(*):
C                   CAM ARRAY WITH CAMPTR AS THE LOAD POINTER
C   ICMPRI(*):      PARALLEL ARRAY TO CAM:  USED TO GENERATE NEXT TOA
C                   FOR DATA MATCHED IN CAM
C   IT:             VALUE OF MIDDLE BITS OF LOCAL TIME--ASTLOC (LOCAL)
C
C
C  START
C
      IF(INL.LT.1) GO TO 30
C  FIND BIN FOR WORD:  IF FULL, DO NOT LOAD
      LCSM=LCBUFF(1,3)/NBIT
      LCSM=MOD(LCSM,NMOD)+1
18    IF(LDCAMP(LCSM).GE.NWPM) GO TO 30
19    LDCAMP(LCSM)=LDCAMP(LCSM)+1
      JJ=LDCAMP(LCSM)
C  WRITE IN BIN
      DO 20 I=1,2
20    LODCAM(LCSM,JJ,I)=LCBUFF(1,I)
      LODCAM(LCSM,JJ,3)=LCBUFF(1,4)
```

```
C       WRITE(6,22)(((LDCAM(ICSM,JJ,I),I=1,3),JJ
22    1 FORMAT(2X,'LDCAM',5(6
        INL=INL-1
C  POP UP LCBUFF
        DO 25 I=1,INL
        DO 25 J=1,4
        II=I+1
25      LCBUFF(I,J)=LCBUFF(II,J)
C  CHECK TIME AVAILABLE TO LOAD
        TLOAD=TLOAD-TIFM
30      CONTINUE
        IF(TLOAD.LT.TIFM) GO TO 50
C  QUANTIZE LOCAL TIME
        LAD=0
        IT=ASILOC*10 +.4+LAD
        IT=IT/NBIT
        IT=MOD(IT,NMOD)+1+IDEL
C  LOAD CAM FROM LODCAM IF LODCAM IS NOT EMPTY
        IF(LDCAMP(IT).LE.0) GO TO 50
36      CAMPTR=CAMPTR+1
        IF(CAMPTR.GT.MAS) CAMPTR=1
        CAMDUA(CAMPTR)=LODCAM(IT,1,1)
        CAMFRQ(CAMPTR)=LODCAM(IT,1,2)
        ICMPRI(CAMPTR)=LODCAM(IT,1,3)
C       WRITE(6,33)(I,CAMDDA(IP),CAMFRQ(JP),ICMPRI(IP),LDCAMP(IT),IP
33      FORMAT(2X,'LD ',4I8)
C  POP UP LODCAM STACK
        IF(LDCAMX(IT).GT.LDCAMP(1)) LDCAMX(IT)=LDCAMP(IT)
        LDCAMP(IT)=LDCAMP(IT)-1
        IIII=LDCAMP(IT)
        DO 35 I=1,IIII
        DO 35 J=1,5
        II=I+1
35      LODCAM(IT,I,J)=LODCAM(IT,II,J)
C  UPDATE LOAD TIME AND START A NEW LOADING IF TIME ALLOWS
        TLOAD=TLOAD-TIFM
        NCWL=NCWL+1
        IF(TLOAD.GT.TIFM) GO TO 5
50      RETURN
        END
```

```
C*****MPPR*****
C*****
      FILE MPPR2 FTN
      SUBROUTINE MPPR
      IMPLICIT INTEGER*4 (I-N)
      REAL MPTLOC
      COMMON/MP/MAMEM(64,4,3),LL,IMOD,IMTAG,IMXF,MAIBUF(64,6),T
     1     NWD
     1   ,IRIT,PN(3),MPNT(3),NIA
      COMMON/AS/MAT IATAG,AIXF,IXF(6),J,TIA,MAS,IWRITE,LAU
      COMMON/PRI/TINC,PRIL,PRIH,NEID,NECON
      COMMON/STAT/MMCNT,MCNT,MAXQ,MAXCAM,MAXMPP,TMAXTQ,NCWL,INL
      COMMON/STA1/   DMAD,ICFO
      COMMON/LMFR/ASXFR(128),LMISIZ,MPTLOC,IBCNT
      COMMON/FM/ CAMFF(64,4),INI
     1    LODCAM(2 ,64,3),LDCAMF(256),LDCAMX(256),
     2    INLT1,INLT2,INLT3


C    THIS ROUTINE ACCEPTS UNMATCHED DATA FOR AQPR, PERFORMS A BETWEEN
C    LIMITS MATCH, TYPES EMITTERS, AND ADDS NEW EMITTERS TO ITS MEMOR
C KEY VARIABLES
C    IMTAG:          PROCESSING FLAG. 1-MORE DATA IN THE INPUT BUFFER TO
C                    TO PROCESSED;     0-A BUFFER ITEM IS IN PROCESS, OK
C                    TO ACCEPT NEW DATA INTO THE BUFFER
C    LMISIZ:         CURRENT SIZE OF THE INPUT BUFFER
C    IBCNT:          NUMBER OF TIMES 'MPPR BUFFER OVERFLOW' FLAG IS SET
C    MAIBUF(LMISIZ,6):   INPUT BUFFER TO THE ARRAY PROCESSORS
C                    1-TOA, 2-FREQ, 3-PW, 4-TOA, 5-ID#, 6-MATCH FLAG
C    ASXFR(LMISIZ):      TRANSFER TIME FOR EACH EMITTER FROM ASSOC
C    NIA             NUMBER OF ARRIVALS TO ARRAY PROCESSORS FOR MATCH
C    MPTLOC:         LOCAL TIME FOR MPPR
C    NECON:          NUMBER OF EMITTERS IDENTIFIED BEFORE CONFIGURATION-
C                    (INPUT PARAMETER)
C    NEID:           NUMBER OF EMITTERS CURRENTLY IN THE ENVIRONMENT-
C                    IF NECON=NEID THEN DYNAMIC CONFIGURATION
C    TINC:           # OF MICRO(INSTRUCTIONS TO PROCESS AN EMITTER IN MPPR
C    TIM:            MICROINSTRUCTION PROCESSING TIME OF THE ARRAY-PROCE
C    KMOD:           DO# FOR A BETWEEN LIMITS MATCH
C    IMOD:           INDEX TO MAIN MEMORY (MODULE NUMBER)
C    LL:             INDEX TO IMOD (POINTER WITHIN A MODULE)
C    MAMEM(IMOD,LL,#)    MAIN MEMORY FILE
C                    #1-TOA
C                    #2-CARRIER FREQUENCY
C                    #3-PULSE WIDTH
C                    #4-TOA
C                    #5-PRI
C                    #6-EMITTER TYPE: 1-REG, 2-PG, 3-CW, 4-JITTERED
C                                     8-UNCLASSIFIED, 9-PG: PRI NOT COMPUTED
C    PNT(#):         POINTER TO CURRENT SIZE OF MEMORY MODULE #
C    MPNT(#):        MAXIMUM SIZE OF MEMORY MODULE #
C    NWD:            NUMBER OF WORDS PER MEMORY MODULE
C    MAXQT:          PROCESSING TIME FOR AN EMITTER (DELAY THROUGH MPPR)
C    MAXQ:           MAXIMUM DELAY FOR PROCESSING AN EMITTER
C    TMXF:           TIME AT WHICH MAXQT OCCURRED
C    TMAXTQ:         TIME AT WHICH MAXQ OCCURRED


C     START
      IF(IMTAG.GE.1)GO TO 28
C  TEST BUFFER FU..
      IF(LMISIZ.LT.64) GO TO 24
      WRITE(6,53)
```

```
23    FORMAT(5X, 'MAP BUFFER OVER-FLOW ')
      IBCNT = IBCNT + 1
      IF(IBCNT .GT. 10) CALL EXIT
      GO TO 30
C  TEST FOR MAP BUFFER SIZE AND ADD INCOMING ELEMENTS TO BUFFER
24    LMISIZ=LMISIZ+1
      IF(LMISIZ.GT.MAXMPP) MAXMPP=LMISIZ+1
      DO 25 I=1,6
25    MAIBUF(LMISIZ,I)    =IXF(I)
30    ASXFR(LMISIZ)=IXF
C COLLECT STAT  ABOUT ARRIVALS TO MPCR
      NIA=NIA+1
C
C   TEST MODULE FOR PROCESSING COMPLETE
      IF(ASXFR(LMISIZ) LE.MPTLOC) RETURN
C
C   TEST FOR # OF EMITTERS IDENTIFIED AND CONFIGURE LIST
      IF(NEID.LT.NRCON) GO TO 28
      NEID=0
      CALL CONFIG
C   TEST FOR LOCAL TIME CURRENT AND INITIALIZE PROCESSING FLAG
28    IMTAG=0
      IF(LMISIZ.LE.1 .AND. MPTLOC.LT.ASXFR(1)) MPTLOC=ASXFR(1)
      IF(LMISIZ.EQ.0) RETURN
C  SEARCH ADJACENT MEMORY MODULES TO MATCH PARAMETERS BETWEEN LIMITS
      TINC=20.*TIM
      KMOD=MAIBUF(1,1)-1
      IF(KMOD.LT.1) KMOD=1
      IF(KMOD.GT.62) KMOD=62
      DO 1 J=1,NMD
      DO 1 IMOD=KMOD,KMOD+2
C  CHECK IF MODULE IS EMPTY (THE TIMING IS 1/3 BECAUSE OF PARALLELISM)
      IF(I.GE.PNT(IMOD)) GO TO 1
C   MATCH IFREQ
      IU=MAMEM(IMOD,I,2)+1
      IL=IU-2
      IF(MAIBUF(1,2).LT.IL) GO TO 1
      IF(MAIBUF(1,2).GT.IU) GO TO 1
      LL=I
      TINC=TINC+(20.*I+5.)*TIM
      GO TO 11
1     CONTINUE
      TINC=TINC+(20.*(I-1))*TIM
      GO TO 12
C   TEST TYPE 1 OR 2 & INCREMENT COUNTER
11    IF(MAMEM(IMOD,LL,6).NE.2 AND.MAMEM(IMOD,LL,6).NE.1) GO TO 61
      IF(MAIBUF(1,1).EQ.IMOD.AND.MAIBUF(1,2).EQ.MAMEM(IMOD,LL,2))
     1 NMCNT=NMCNT+1
      IF(MAIBUF(1,1).NE.IMOD) IDOAD=IDOAD+1
62    IF(MAIBUF(1,2).NE.MAMEM(IMOD,LL,2)) ICFD=ICFD+1
      IF(MAIBUF(1,1).NE.IMOD) GO TO 64
      IF(IWRITE.LT.1) GO TO 64
      WRITE(6,403) IP
403   FORMAT(1X,'CAM POINTER IS',I3)
      DO 43 K=1,NMOD
      II=LDCAMP(K)
      DO 43 I=1,II
43    WRITE(6,406) K,(LDDCAM(K,I,J),J=1,3),I
406   FORMAT(2X,'LDDCAM',5I6)
64    CONTINUE
C   TEST FOR TYPE DECOD AND INCLUDE TIMING FOR INTERRUPT CHECK
61    TINC=TINC+3.*TIM
      IF(M.EQ.1) TINC=TINC+10.*TIM
```

```
      TINC=TINC+3.*TIM
C   *COMPUTE IPRI
      CALL PRIF
C   TEST FOR CLASSIFICATION INCOMPLETE
      IF(MAMEM(IMOD,LL,6).EQ.4 AND MAMEM(IMOD,LL,7) LE.4, GO TO 55
C   TEST FOR PROPER MODULE
10    IF(MAIBUF(1,1) EQ (MOD) GO TO 65
C   UPDATE CORRECT MODULE
      CALL UPDAM
      TINC=TINC+60.*TIM
      GO TO 19
C   UPDATE CURRENT MODULE
55    CALL UPDN
      TINC=TINC+60.*TIM
      GO TO 19
C   INSERT UNMATCHED DATA IN PROCESSOR FILE
12    TINC=TINC+40.*TIM
      K=MAIBUF(1,1)
      N=PNT(K)
      MAMEM(K,N,1)=1
      MAMEM(K,N,2)=MAIBUF(1,2)
      MAMEM(K,N,3)=MAIBUF(1,3)
      MAMEM(K,N,4)=MAIBUF(1,4)
      MAMEM(K,N,5)=-1
      MAMEM(K,N,6)=2
      MAMEM(K,N,7)=0
      MAMEM(K,N,8)=MAIBUF(1,5)
      PNT(K)=PNT(K)+1
      IF(IWRITE.GT.0)WRITE(6,15)
15    FORMAT(5X'NO MPP MATCH')
      IF(PNT(K).GT.MPNT(K)) MPNT(K)=PNT(K)-1
      IF(MPNT(K).LE.NWD) GO TO 19
      WRITE(6,99)
99    FORMAT (5X,'MPPR FILE FULL')
C
      RETURN
C
C   UPDATE MPTLOC
19    MPTLOC=MPTLOC+TINC
C   SET PROCESSING COMPLETION TIME
75    TMXF=MPTLOC
C   COMPUTE DELAY IN PROCESSING
      AXQT=MPTLOC*10.**6-MAIBUF(1,4)
      IF(IWRITE.GT.0)WRITE (1,400)AXQT,MPTLOC,MAIBUF(1,4)
400   FORMAT(2X,'AXQT= ',E15.6,E15.5,I9)
      AXQT=AMOD(AXQT,2.**15)
      MAXQT=AXQT
      IF(MAXQT.GT.MAXQ)TMAXTQ=TMXF
      IF(MAXQT.GT.MAXQ) MAXQ=MAXQT
      IF(IWRITE.GT.0)WRITE(6,6)(MAIBUF(1,I),I=1,3),MAXQT,LMISIZ,MPTLOC
     1  ,MAIBUF(1,5)
6     FORMAT(1X,'MPPR',5I8,3X,E14.4,I4)
C  COLLECT STAT. FOR LCAM BUFFER
      IF(INL.GT.INLM) INLM=INL
C   POP UP STACK
      LMISIZ=LMISIZ-1
C
      IF(LMISIZ.LT.1) RETURN
C
      DO 80 I=1,LMISIZ
      K=I+1
      ASXFR(I)=ASXFR(K)
      DO 80 J=1,4
```

```
      MAIBUF(I)=MAIBUF(N-I)
50    CONTINUE
C     TEST FOR PROJECTION COMPLETE
61      IF(MPTLOC.GE.MSAFR(LMISIZE).AND.LMISIZE.LE.1) IMTAG=1
        IF(MPTLOC.LT.MSAFR(1))MPTLOC=ADATM(1)
100     RETURN
        END
```

```
C*****PRIP*****************************************************************
C**************************************************************************
C**************************************************************************
      SUBROUTINE PRIP
C     IMPLICIT INTEGER*4 (I-N)
      COMMON/MP/MAMEM(64,40,8),LL,IMOD,IMTAG,IMXF,MAIBUF(64,6),TIM
     1    ,IWD
     2    ,IPST,PN(64),MPNT(64),NIA
      COMMON/PRI/TINC,PRIL,PRIH,NEID,NECON
C
C
C   THIS ROUTINE COMPUTES THE PRI'S (IN THE MICROPROCESSOR ARRAY
C   MEMORY) AND TESTS FOR EMITTER TYPE AND MISSED PULSES FOR THOSE
C   DATA WORDS MATCHED IN MFPR
C
C   KEY VARIABLES
C   IMOD:              INDEX TO MEMORY MODULE
C   LL:                INDEX TO MATCHED WORD
C   MAIBUF(1,#)    #1-DOA; #2-FREQ; #3-PW; #4-TOA; #5-ID#; #6-MATCH FLAG
C   MAMEM (IMOD,LL,1):   DOA
C         (IMOD,LL,2):   CARRIER FREQUENCY
C         (IMOD,LL,3):   PULSE WIDTH
C         (IMOD,LL,4):   TOA
C         (IMOD,LL,5):   PRI
C         (IMOD,LL,6):   EMITTER TYPE (1-REG, 2-PG, 3-CW, 4-JITTERED,
C                        8-UNCLASSIFIED, 9-PG: PRI NOT DETERMINED)
C         (IMOD,LL,7):   NUMBER OF PULSES RCVD, UP TO 4
C         (IMOD,LL,8):   EMITTER ID NUMBER
C   TINC:       TOTAL # OF MICROINSTRUCTIONS EXECUTED TO PROCESS
C               AN EMITTER IN MFPR
C   TIM         MICROINSTRUCTION PROCESSING TIME OF THE PROCESSOR
C               ARRAY
C   IPRI:       PULSE REPETITION INTERVAL  (LOCAL TO PRIP)
C
C
C START
      IF(MAMEM(IMOD,LL,4).LT.0) RETURN
C
C   COMPUTE PRI(MOD 2**16)
      IPRI=MAIBUF(1,4)-MAMEM(IMOD,LL,4)
      IF(IPRI.LT.0)IPRI=IPRI+2**14
C   TEST FOR PULSE GROUPS
      TINC=TINC+15.*TIM
      IF(IPRI.GT.50) GO TO 45
      MAMEM(IMOD,LL,6)=9
      TINC=TINC+4.*TIM
      GO TO 49
C   TEST FIRST PULSE
45    TINC=TINC+TIM*4.
      IF(MAMEM(IMOD,LL,5).LT.0) GO TO 100
C   TEST FOR REGULAR EMITTER
      IPFDUM=MAMEM(IMOD,LL,5)
       I1234=IPRI-IPFDUM
      IX=IABS(I1234)
      TINC=TINC+14.*TIM
      IF(IX.LT.3) GO TO 12
C   TEST FOR JITTERED PRI
      TINC=TINC+TIM*6
      IF(MAMEM(IMOD,LL,7).LT.4) GO TO 40
      MAMEM(IMOD,LL,6)=4
      TINC=TINC+4.*TIM
      GO TO 49
C    TEST FOR PULSE GROUPS
```

```
12      TINC=TINC+5.*TIM
        IF(MAMEM(IMOD,LL,6).NE.9) GO TO 13
        MAMEM(IMOD,LL,6)=2
        GO TO 14
13      MAMEM(IMOD,LL,6)=1
14      TINC=TINC+4.*TIM
        GO TO 49
C     TEST PRI COUNT
40      TINC=TINC+TIM*4
        IF(MAMEM(IMOD,LL,7).EQ.0) GO TO 100
C   TEST SINGLE MISSED PULSE
        X=ABS(2.*MAMEM(IMOD,LL,5)-IPRI)
        XA=ABS(2.*IPRI-MAMEM(IMOD,LL,5))
        TINC=TINC+2.*TIM
        IF(X.GT.3.) GO TO 30
        WRITE(6,17)
17      FORMAT(1X,'MISSED PULSE')
        TINC=TINC+3.*TIM
        GO TO 49
30      TINC=TINC+TIM*3
        IF(XA.GT.3.) GO TO 100
        IF(IRIT.GT.0)WRITE(6,17)
        MAMEM(IMOD,LL,5)=IPRI
        GO TO 49
C     STORE CURRENT PARAMETERS
100     TINC=TINC+4.*TIM
        MAMEM(IMOD,LL,5)=IPRI
        MAMEM(IMOD,LL,7)=MAMEM(IMOD,LL,7)+1
C       IF(IRIT.GT.-1)WRITE(6,16)MAMEM(IMOD,LL,7),IPRI
16      FORMAT(1X,I2,2X'PRI COMPUTED',I4)
        TINC=TINC+TIM*15
        IF(IPRI.LT.PRIL)   PRIL=IPRI
        IF(IPRI.GT.PRIH)   PRIH=IPRI
        NEID=NEID+1
49      MAMEM(IMOD,LL,4)=MAIBUF(1,4)
50      RETURN
        END
```

```
C*****UPDAAM*****
C*****
      SUBROUTINE UPDAAM
C     IMPLICIT INTEGER*4 (I-N)
      COMMON/MP/MAMEM(64,46,8),LL,IMOD,INTAG,TMXF,MAIBUF(64,6),TIM,
     1     NWD
     1   ,IRIT,PNT(64),MPNT(64),NIA
      COMMON/PM/LCBUFF(64,4),INL
     1    ,LODCAM(256,64,3),LOCAMP(256),LOCAMX(256),
     2     INLT1,INLT2,INLT3
C
C
C  THIS ROUTINE IS USED TO UPDATE THE CONTENTS OF THE
C  MICROPROCESSOR ARRAY INTERNAL MEMORY WHEN THE DATA
C  CHANGES WITHIN THE ALLOWED LIMITS AND THE DOA ALSO
C  CHANGES, I.E., THE MPPR MATCH OCCURRED IN AN ADJACENT
C  DOA MODULE.
C
C  KEY VARIABLES:
C   PNT(KK):       POINTER TO THE CURRENT SIZE OF MODULE KK OF THE
C                  DATA FILE MAMEM(KK,*,*)
C   MPNT(KK):      MAXIMUM VALUE OF PNT(KK)
C   MAIBUF(1,#): #1-DOA, #2-FREQ, #3-PW, #4-TOA, #5-ID#, #6-MATCH FLAG
C   LCBUFF(INL,*): #1-DOA, #2-FREQ, #3-NTOA, #4-PRI
C   MAMEM(IMOD,LL,#):
C                  #1-DOA, #2-FREQ, #3-PW, #4-TOA, #5-PRI (SET TO -1
C                  UNTIL COMPUTED), #6-EMITTER TYPE (SEE PRIF),
C                  #7-# PULSES RCVD, #8-EMITTER ID#
C   IMOD:          MEMORY MODULE WHICH CORRESPONDS TO DOA (1..61)
C   LL:            EMITTER LOCATION IN MEMORY, POINTS TO THE LLTH EMITTER
C                  STORED IN THE IMODTH SECTION
C
C
C START
      KK=MAIBUF(1,1)
      JJ=PNT(KK)
      DO 2 J=1,4
2     MAMEM(KK,JJ,J)=MAIBUF(1,J)
      DO 3 K=5,7
3     MAMEM(KK,JJ,K)=MAMEM(IMOD,LL,K)
      MAMEM(KK,JJ,8)=MAIBUF(1,5)
      PNT(KK)=PNT(KK)+1
      IF(PNT(KK).GT.MPNT(KK)) MPNT(KK)=PNT(KK)-1
C   STORE MATCHED DATA IN NEXT ARRIVAL TIME ARRAY
      INL=INL+1
      INLT2=INLT2+1
      IF(INL.LE.64) GO TO 22
      WRITE(6,23)TMXF
23    FORMAT(2X,'FMNAT INPUT BUFFER OVERFLOW.  MT=',F12.6)
      GO TO 49
22    LCBUFF(INL,1)=MAIBUF(1,1)
      LCBUFF(INL,2)=MAIBUF(1,2)
      LCBUFF(INL,3)=MAMEM(IMOD,LL,5)+MAIBUF(1,4)
      LCBUFF(INL,4)=MAMEM(IMOD,LL,5)
      IF(IRIT .GT. 0) WRITE(6,26)(LCBUFF(INL,JJ),JJ=1,4),INL,TMXF
26    FORMAT(2X,'MTRA',5I8,F12.6)
C  PACK MEMORY
49    PNT(IMOD)=PNT(IMOD)-1
      NN=PNT(IMOD)
      DO 50 J=1,8
50    MAMEM(IMOD,LL,J)=MAMEM(IMOD,NN,J)
      MAMEM(IMOD,NN,8)=0
      RETURN
```

```
C*****UPDM*****
C*****
      SUBROUTINE UPDM
C     IMPLICIT INTEGER*4 (I-N)
      COMMON/MP/MAMEM(64,4,8) LL,IMOD,INTAG,TMXF,MAIBUF(64,6),TIM,
     1    NWD
     2   ,IRIT,PNT(64),MPNT(64),NIA
      COMMON/FM/LCBUFF(64,4),INL,
     1   LMODAM(256,64,3),LCAMP(256),IDAMX(256),
     2     INLT1,INLT2,INLT3

C
C
C  THIS ROUTINE IS USED TO UPDATE THE CONTENTS OF THE
C  MICROPROCESSOR ARRAY INTERNAL MEMORY WHEN DATA CHANGES
C  WITHIN THE ALLOWED LIMITS BUT THERE IS NO CHANGE IN DOA.
C
C  KEY VARIABLES:
C    MAIBUF(1,#):   #1-DOA, #2-FREQ, #3-PW, #4-TOA, #5-ID#, #6-MTCH FLAG
C    INL:           POINTER TO SIZE OF LCBUFF
C    LCBUFF(INL,#):  #1-DOA, #2-FREQ, #3-NTOA, #4-PRI
C    MAMEM(IMOD,LL,#):
C                   #1-DOA, #2-FREQ, #3-PW, #4-TOA, #5-PRI, #6-TYPE,
C                   #7-PULSES RCVD, #8-EMITTER ID#
C    TINC:          TOTAL # OF MICROINSTRUCTIONS EXECUTED TO PROCESS
C                   AN EMITTER IN MPPR
C    TIM:           MICROINSTRUCTION PROCESSING TIME OF THE PROC ARRAY
C
C
C START
      IF(IRIT.GT.0)WRITE(6,10)MAMEM(IMOD,LL,6)
10    FORMAT(5X,'MPP MATCH    TYPE='I2)
      KK=MAIBUF(1,1)
      DO 2 J=1,4
2     MAMEM(KK,LL,J)=MAIBUF(1,J)
C   STORE MATCHED DATA IN MAT ARRAY
      INL=INL+1
      INLT3=INLT3+1
      IF(INL.LE.64) GO TO 22
      WRITE(6,23)TMXF
23    FORMAT(2X,'FMMAT INPUT BUFFER OVERFLOW.  MT=',F12.6)
      GO TO 49
22    LCBUFF(INL,1)=MAIBUF(1,1)
      LCBUFF(INL,2)=MAIBUF(1,2)
      LCBUFF(INL,3)=MAMEM(IMOD,LL,5)-MAIBUF(1,4)
      LCBUFF(INL,4)=MAMEM(IMOD,LL,5)
      TINC=TINC+12.+TIM
      IF(IRIT .GT.0) WRITE(6,26)(LCBUFF(INL,JJ),JJ=1,4),INL,TMXF
26    FORMAT(2X,'MTR',5I8,F12.6)
49    RETURN
      END
```

```
C*****CONFIG*****
C*****
C  FILE CONFIGURATION
      SUBROUTINE CONFIG
C     IMPLICIT INTEGER*4 (I-N)
      COMMON/PRI/TINC,PRIL,PRIH,NEIO,NECOM
      COMMON/LABMP/PCXFR(100),LAIS(2),ASTLOP,ASTLOP,IDEL,NMOD,
     1  NWPM,NBIT
      DIMENSION M(4,13)
C
C
C                ------PRIL----  ---PRIH-----  NMOD  NBIT
      DATA M/     176,   896,   3000,  3044,    32,   64,
     1            128,   640,   2048,  3072,    64,   64,
     2            128,   256,   3072,  4096,    64,   64,
     3            128,   256,   4096,  8001,    64,  128,
     4            256,   768,   4096,  7168,    64,  128,
     5            768,  1000,   4096,  5118,    64,  128,
     6            256,   896,   7168,  8001,    32,  256,
     7            768,  1024,   5118,  7168,    32,  256,
     8           1000,  1024,   4096,  5118,    32,  256,
     9            896,  1025,   7168,  8001,    16,  512,
     1            256,  1024,   3072,  4096,    32,  128,
     1            640,  1024,   2048,  3072,    32,  128,
     2            896,  1024,   2000,  2046,    32,  128
     3       /
C
C   THIS SUBROUTINE SIMULATES THE ROM WHICH MAPS THE
C   MAX. AND THE MIN. PRI'S INTO THE CONFIGURATION
C
C   KEY VARIABLES:
C     NMOD:     NUMBER OF MODULES IN THE LIST BUFFER (LCBUFF)
C     NBIT:     NUMBER OF BITS SHIFTED
C     PRIL,PRIH:    LOW AND HIGH PULSE REPETITION INTERVAL
C
C
C START
      NCONF=13
      DO 50 I=1, NCONF
        IF(PRIL.LT.M(1,I) .OR. PRIL.GE.M(2,I)     .OR.
     1     PRIH.LT.M(3,I) .OR. PRIH.GE.M(4,I)) GO TO 50
        NMOD=M(5,I)
        NBIT=M(6,I)
        RETURN
50    CONTINUE
      NMOD=16
      NBIT=128
      RETURN
      END
```

```
      SUBROUTINE WRITAT
C     IMPLICIT INTEGER*4 (I-N)
      REAL MPTLOC
      INTEGER CAMPTR,CAMPRA,CAMFRQ,CAMPTR,CAMSELF
      COMMON/EMIT/ARRIR(500,25),TIME,VP,VR,PCONST,NE,MINPTR,
     1  EMITON(500),TMIN
      COMMON/RCV/I,M4,IFREQ,IPW,ITOA,INUM,SL(100),A,B,IRTAG,RTXF,J,IQ
      COMMON/AE/NA,IATAJ,ATXT,IXF(6),JTA,TIA,NAS,IWRITE,L40
      COMMON/MP/MAMEM(64,40,8),LL,JMOD,IMTAG,TMXF,MAIBUF(64,6),TTT,
     1    NWD
     1  ,IRIT,PN,(64),MPNT(64),NIA
      COMMON/UP/CAMPRA(64),CAMFRQ(64),LMPRI(64),CAMPTR
      COMMON/D4/LM
      COMMON/PRI/IPRC,PRIL,PRIH,NEID),NECUN
      COMMON/UP3/CAMBUF(32,5)
      COMMON/LABUC/ISF,TLOAD,TIFM,NEA
      COMMON/STAT/MMCNT,MCNT,MAXQ,MAXCAM,MAXMPP,TMAXTQ,NCWL,INLM
      COMMON/STAT1/IDOAD,ICFD
      COMMON/RAND/IRAN,JRAN
      COMMON/LRCVR/PCBUF(64),LRIBUF(64,5),IACELL(16),ACELL(16),
     1  RID(64,6),TRIB(64),LRIBT,RCTLOC,LRISIZ,LRIT
      COMMON/LABPR/RCXFR(32),LAISIZ,ASTLOC,ASTLOP,IDEL,NMOD,
     1  NWPM,NBIT
      COMMON/LMPPR/AQXFR(128),LMISIZ,MPTLOC,IBCNT
      COMMON/PM/LCBUFF(64,4),INL
     1  ,LDCAM(256,64,3),LDCAMP(256),LDCAMX(256),
     2  INLT1,INLT2,INLT3
      COMMON/CMAIN/TPRINT,PINC,RTIME,PTIME,VR
      COMMON/JMPZER/JICW,JIPQ,J7FR,JIDOA,JZPRI
C
      IF(TIME.LT.TPRINT) RETURN
      TPRINT=TPRINT+PINC
C
C
C*********************  WRITE OUT STATISTICS  ***********************
C
      WRITE(6,77) TIME,PINC
77    FORMAT(////' TIME =',F6.2,' SEC.    (PRINT INCR=',F7.3,')')
101   WRITE(6,102) MMCNT,MCNT,MAXQ,MAXCAM,MAXMPP,TMAXTQ,NCWL,INLM
     1  ,IDOAD,ICFD,NEA,NIA,INLT1,INLT2,INLT3
102   FORMAT(/,1X,'MMCNT=',I9,2X,'MCNT=',I9,2X,'MAX=',I7,2X,'CAMB=',I3,2X
     1  'MPPB=',I3,/,2X,'MTIME=',F9.6,3X'NCWL=',I9,
     2  2X,'MLCAMB=',I3,2X,'IDOAD=',I7,/,2X,'ICFD=',I5,2X,'NEA=',I9,
     3  2X,'NIA=',I7,2X,'INLT1=',I4,2X,'INLT2=',I4,2X,'INLT3=',I5)
      ITMPNT=0
      DO 108 K=1,64
108     ITMPNT=ITMPNT+PNT(K)-1
      WRITE(6,111)(MPNT(I),I=1,64)
111   FORMAT(2X,16I3)
      WRITE(6,1004)(IFIX(PNT(I)-1),I=1,64)
1004  FORMAT(' MAIN MEMORY SIZES:'/(2X,16I3))
      WRITE(6,115) ITMPNT
115   FORMAT(5X,'TOTAL EMITERS IN MPPR=',I3)
      WRITE(6,109)(LDCAMX(I),I=1,NMOD)
109   FORMAT(2X,16I3)
      LCS2PT=0
      DO 1008 I=1,NMOD
        LCS2PT=LCS2PT+LDCAMP(I)
1008  CONTINUE
      WRITE(6,1009)(LDCAMP(I),I=1,NMOD)
1009  FORMAT(' LDC SIZES:'/(2X,16I3))
      WRITE(6,1010)LCS2PT,FLOAT(LCS2PT)/NMOD
1010  FORMAT(' TOTAL LIST SIZE=',I4,'  AVG LIST SIZE=',F6.2)
```