AD-A070 848   GOODYEAR AEROSPACE CORP AKRON OHIO   F/G 9/2
MICROCOMPUTER ARRAY PROCESSOR.(U)
OCT 78 R H RIES, R A HUJAR, F C CARTY   F33615-75-C-1179
UNCLASSIFIED   GER-16565   AFAL-TR-78-157   NL

1 OF 2
AD
A070848

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
18 AFAL-TR-78-157 ✓

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
6 MICROCOMPUTER ARRAY PROCESSOR.

5. TYPE OF REPORT & PERIOD COVERED
9 FINAL rept.
2 June 1975 - 2 August 1978

6. PERFORMING ORG. REPORT NUMBER
14 GER-16565 ✓

7. AUTHOR(s)
10 R.H. Ries, R.A. Hujar, F.C. Carty e
M.H. Diehl

8. CONTRACT OR GRANT NUMBER(s)
15 F33615-75-C-1179 New

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Goodyear Aerospace Corporation ✓
1210 Massillon Road
Akron, Ohio 44315

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Project 7633

12 139 p.

11. CONTROLLING OFFICE NAME AND ADDRESS
Air Force Avionics Laboratory (WRP)
Air Force Systems Command
Wright-Patterson AFB, Ohio 45433

12. REPORT DATE
11 October 1978

13. NUMBER OF PAGES
118

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
NONE

16. DISTRIBUTION STATEMENT (of this Report)

16 7633

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
ESM DATA PROCESSING
MULTIPROCESSOR
MULTIPLE MICROCOMPUTER SYSTEM
MICROCOMPUTER ARRAY PROCESSOR
PARALLEL ARRAY PROCESSOR

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The objective of this work was to design, fabricate, and bench test a feasibility model of an EW computer architecture based on utilizing multiple microprocessors in a multiprocessor system. The developed model consists of four microprocessors integrated into a tightly coupled nearly symmetrical structure exhibiting a master-slave relationship among its processors. Each microprocessor is composed of a 32-bit CPU and a dedicated local

Over page

156 800

JOB

program memory. Instruction execution times range from 250 to 600 nsec dependent upon instruction type. The four CPU effect interprocessor communication through an interrupt structure and message switching via global memory which is shared by all processors. The global memory is divided into three independent banks which support a maximum transfer rate of 15 million words a second. Each bank solves the memory contention problem by queueing up its request and then servicing processors within the queue on a priority basis.

The function of the multiprocessor is to sort pulse trains based on digital pulse intercepts collected by a wide-open channelized receiver. Once the multiprocessor determines the PRI of an emitter, all emitter parameters are passed to a preprocessor which is inserted in the data stream between the receiver and the multiprocessor. The function of the preprocessor is to remove from the data stream all pulses from emitters identified by the multiprocessor. The feasibility model preprocessor can accept a peak receiver output pulse rate of 340,000 pulses per second.

FOREWORD

This final report was prepared by Goodyear Aerospace Corporation, Akron, Ohio, under USAF contract F33615-75-C-1179, Project 7633, entitled "Microcomputer Array Processor". The contract was initiated by the Air Force Avionics Laboratory, Air Force Wright Aeronautical Laboratories, Wright Patterson Air Force Base, Ohio. Mr. Joseph Caschera, Electronic Warfare Division, AFAL/WRP, is the Air Force Project Engineer. This report covers the period 2 June 1975 to 2 August 1978 and was submitted by the authors on 23 August 1978.

The Goodyear Aerospace personnel involved in this program and in the writing of this report are R. H. Ries (project engineer), F. G. Carty, M. D. Diehl, R. A. Hujar and M. J. Kroeger. The contractor's report number is GER-16565.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

INTRODUCTION

1. SCOPE

This final technical report describes the work performed on the Microcomputer Array Processor development program under contract F33615-75-C-1179. The purpose of the 36 month program was to design and fabricate a feasibility model of an electronic warfare (EW) processor based on utilizing the latest available large-scale integration (LSI) microcomputer technology in a multiprocessor system architecture. This report will discuss the functional operation of the fabricated equipment and its hardware characteristics.

2. BACKGROUND

The overall goal of this effort is the conceptual development of an EW data processing subsystem capable of sorting, identifying and tracking emitter signals in real time, on a pulse-by-pulse basis, for the very dense radar environments. This effort seeks the needed increased computer capability through utilization of emerging LSI technology in the form of multiple microprocessors. The various microcomputers are organized into a multiprocessor system architecture which effects throughput improvement through concurrency of operation of its individual processors.

Work directed towards this goal was begun under contract F33615-74-C-1101 entitled ESM HYBRID PROCESSING TECHNIQUES (HPT). This program investigated both microcomputer chip architectures and multiprocessor system architectures in order to:

1) Establish the architectural features which must be possessed by a microcomputer chip to make it suitable for EW type data processing,

2) Determine support circuitry required to develop a full capability microprocessor based on the microcomputer chip, and

3) Develop a conceptual multiprocessor system architecture which would facilitate the concurrent utilization of multiple processors on a given EW problem.

A computer simulation of the conceptual design was also developed and extensively tested to optimize the multiprocessor design and predict its performance in an EW application. Details of this work are documented in the ESM HYBRID PROCESSING TECHNIQUES final report (AFAL-TR-75-125) and will not be repeated here. During the initial efforts on Phase I of the current Contract the conceptual design was translated into a detailed hardware design. The translation step also involved some architectural refinement in order to make optimum use of the integrated circuit technology available at the time of fabrication. Details on the Phase I activities may be found in Phase I Microcomputer Array Processor Interim Report dated November 1976.

Under Phase II of the MAP contract a feasibility model of the hardware design was fabricated and checked out. The final phase (Phase III) integrated the fabricated model with a channelized receiver and developed the necessary software to process intercepts from the receiver. An overview of the fabricated system is covered in the next item.

3. FEASIBILITY MODEL PROCESSOR OVERVIEW

A block diagram of the fabricated data processor is shown in Figure 1. The input to the processing system consists of digitally encoded radar pulses intercepted by the receiver subsystem. The functional responsibility of the data processor is to establish, track and report pulse trains based on the raw radar intercepts outputted by the receiver. In order to achieve the extremely high processing rates needed to perform these tasks, the processing is partitioned among the various subsystems which operate concurrently as shown in Figure 1. Here the processing tasks of tracking, emitter establishment and display processing are partitioned among the preprocessor, multiprocessor and display processor respectively. A detailed discussion of each of these subsystems appear in the following sections. Section II describes the preprocessor, Section III the multiprocessor and Section IV the display processor. In addition, Appendix A contains detailed flowcharts and program listings for the preprocessor and the multiprocessor. Appendix B describes the communication structure between the preprocessor and the multiprocessor.

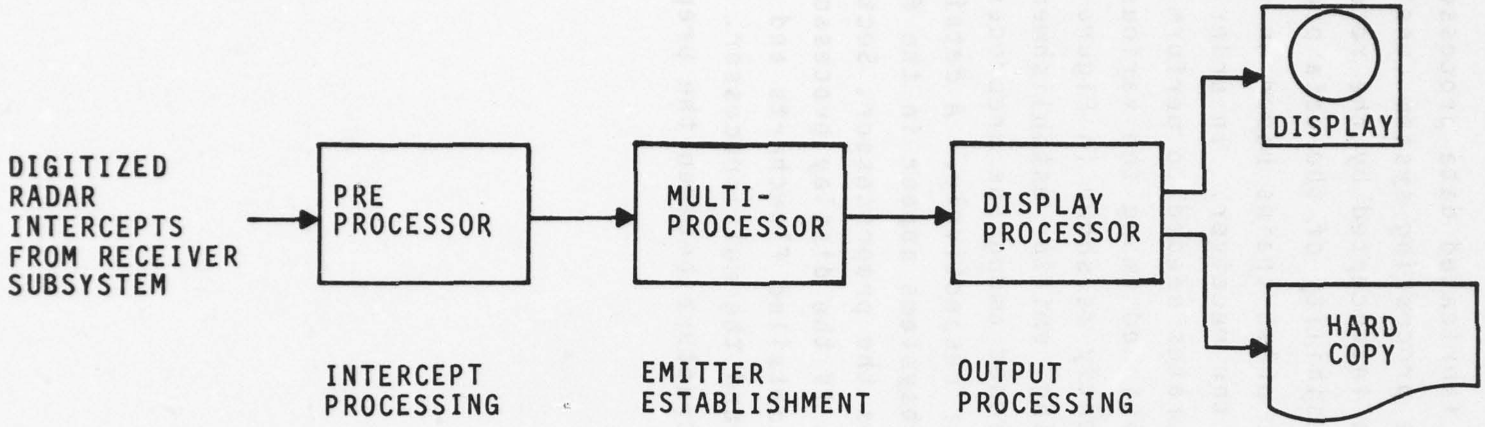


Figure 1 - Passive Detection System

SECTION II

THE PREPROCESSOR

1. FUNCTIONAL OVERVIEW

A block diagram of the preprocessor is shown in Figure 2. The functional responsibility of this hardware is to accept the digitized radar pulse intercepts outputted by the receiver and correlate each intercept against an established emitter file for pulse train tracking and data filtering operations. The objective is to reduce the data rate into the multiprocessor by filtering, from the input pulse stream, those intercepts that originate from emitters which are currently being tracked by the preprocessor. This data rate reduction is essential to allow handling of a very high receiver data rate while still maintaining sufficient processing time per radar intercept in the multiprocessor to execute complex PRI establishment algorithms.

The filtering via correlation operation consists of comparing the parameters of each intercept pulse against the parameters of the emitter words stored in the preprocessor memory. Hardware hash addressing techniques are used to select that subset of the emitter file over which a particular correlation operation could be meaningful. If the intercept matches the emitter word within predetermined tolerances for each selected parameter, correlation is said to occur. The parameter comparisons performed are given in equations 1 through 5.

$$F_E+\Delta_F \geq F_I \geq F_E-\Delta_F, \tag{1}$$

$$PW_E+\Delta_P \geq PW_I \geq PW_E-\Delta_P, \tag{2}$$

$$AOA_E+\Delta_A \geq AOA_I \geq AOA_E-\Delta_E, \tag{3}$$

$$LTOA_E+PRI+\Delta_T \geq TOA_I \geq LTOA+PRI-\Delta_T, \tag{4}$$

or if $(TOA_I \geq LTOA_E+PRI+\Delta_T)$ then

$$LTOA_E+2PRI+2\Delta_T \geq TOA_I \geq LTOA+2PRI-2\Delta_T, \tag{5}$$

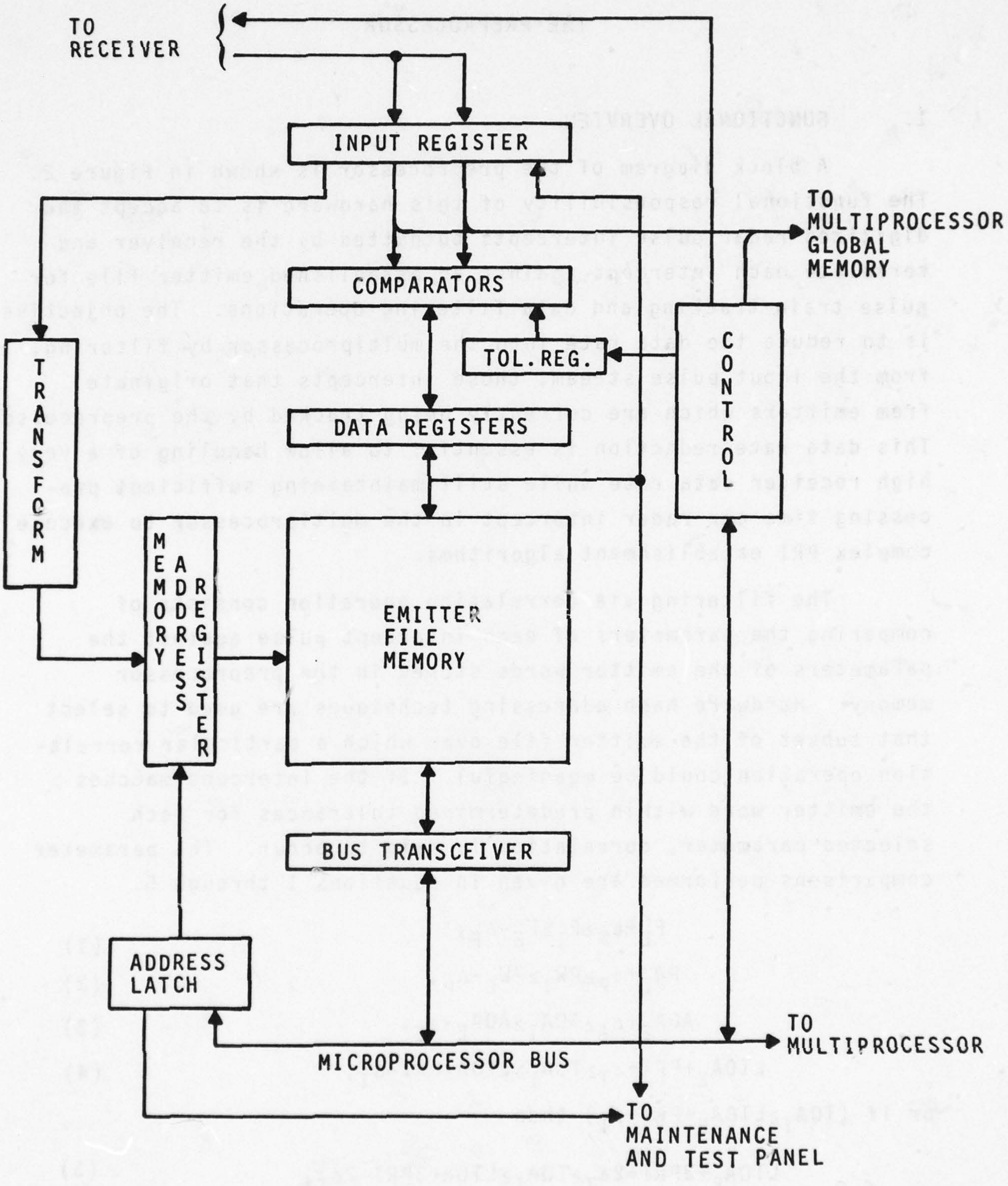


Figure 2 - Preprocessor Block Diagram

where

F = Radio Frequency,

PW = Pulse Width,

AOA = Angle of Arrival

TOA = Clock time of Arrival of the intercept as measured at the receiver,

LTOA = Last known time of arrival stored in the emitter file for a particular emitter, and

Δ = Tolerance associated with each emitter parameter.

and the subscripts are:

E = Refers to parameters of the stored emitter word, and

I = Refers to parameters of the input intercept.

The two different time tests are used so that alternate pulses of a given emitter can be missed by the receiver without losing track of the emitter. However, if a number of consecutive pulses are missing the LTOA value soon becomes to old in time to be of any value for correlation. When this occurs a limited number of attempts are made to resync on the pulse train.

When correlation occurs, parameters in the pulse intercept word are used to update the emitter file word in order to keep the emitter word parameters current. At this point, no further processing of the intercept word is required and it is dropped from the data stream passed on to the multiprocessor.

When correlation does not occur, it is assumed that the pulse word represents a new (yet unrecognized) emitter. In this case the preprocessor does not eliminate the intercept from further system consideration, but sends it on to the intercept buffer of the multiprocessor subsystem for pulse train analysis.

2. DETAILED PREPROCESSOR OPERATION

a Input Processing

Both the receiver and preprocessor share a common characteristic in that their respective peak throughput rates are substantially different from their average throughput rates. This dynamic variation in instantaneous behavior may be linked to different causes in the radar environment making it unlikely that the two throughput rates will vary in unison. Thus, a buffer is inserted between the two devices to decouple their instantaneous rates and allow each to operate asynchronously. The preprocessor communicates with this buffer over a 15 foot twisted pair cable containing 36 unidirectional signals. Limited multiplexing is used to reduce the bus interface cost. The interface signal consists of the following:

.Signals from buffer to Preprocessor

1) Data lines - 32 signals
2) Data ready line - 1 signal

.Signals to buffer from Preprocessor

1) Transfer complete
2) Reset
3) Hi/Lo halfword - 1 signal
4) Buffer Inhibit - 1 signal

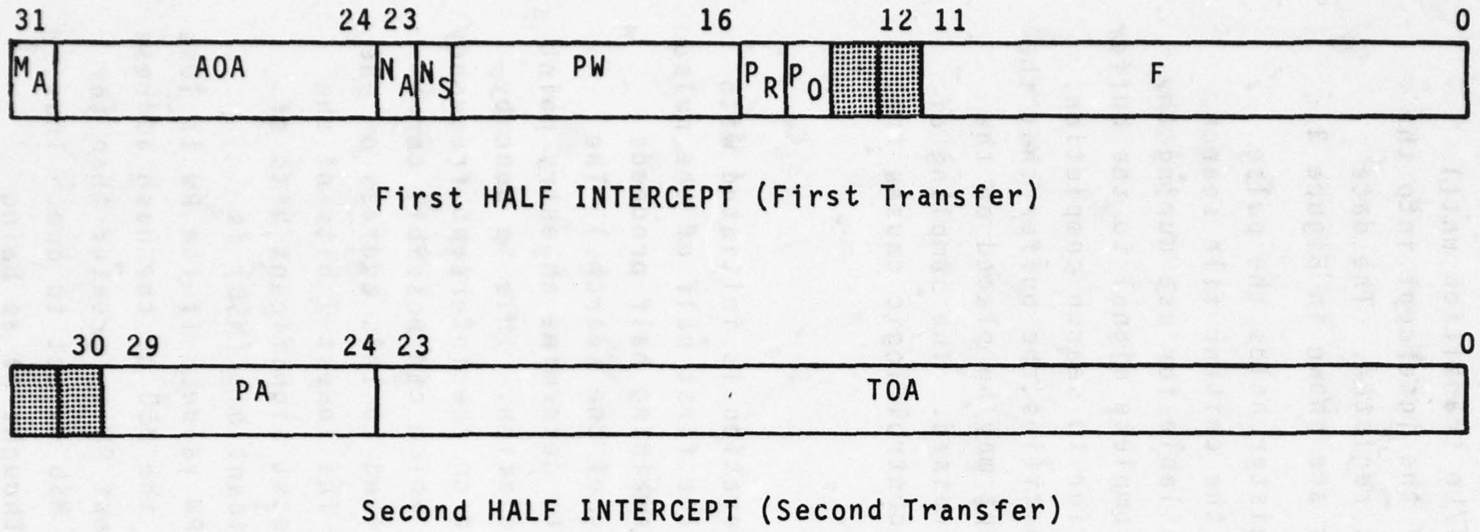
Note: all lines are differentially driven and received.

Transfer of a pulse intercept is initiated when the data ready line to the preprocessor is asserted. This line is periodically sampled by the preprocessor microprogram control logic and 153 n.sec. after assertion is detected, the information on the 32 data lines is clocked into the preprocessor input register. This operation effects transfer of the first half of the pulse intercept from the receiver buffer. One clock pulse (i.e., 153 n.sec.) later the Hi/Lo signal is changed by the preprocessor, thereby requesting the second half of the intercept. A total of 600 n.sec. is allowed for signal propagation from the Hi/Lo transition until clocking of the second half of the intercept into the remaining portion of the input register. The data format for these two transfers are shown in Figure 3.

The preprocessor input register holds the pulse intercept until completion of the emitter file search. Since this register is not available for use during the search operation, a transfer complete signal to the buffer is not initiated until just prior to search completion. The transfer complete signal notifies the buffer that the first half of the next intercept may be placed on the data lines and the data ready raised. The sampling of this line by the microprogram control logic causes the entire input process to repeat.

b Search and Correlation

The emitter file search operation is initiated with completion of the transfer of the first half of the pulse intercept. (Transfer of the remaining half proceeds concurrently with the beginning of the search.) The initial step of the search is to determine an entry point into the emitter file for correlation. This is done by developing a file address based on the intercept frequency and pulse width. All emitters which can possibly correlate with the intercept are linked to this address or one which can be derived from it. The lowest 7 bits of the intercept frequency form the least significant bits of this address. The most significant bit (MSB) is derived from a break point in PW value. If the PW is less than this predefined constant, the MSB of the hash address is set to zero. If the intercept PW is greater than (or equal to) this break point the MSB is set to one. Thus, the file entry address may be thought of as being



First HALF INTERCEPT (First Transfer)

Second HALF INTERCEPT (Second Transfer)

M_A = Multiple AOA Flag
N_A = No AOA data
N_S = No slot data
PA = Pulse Amplitude
F = Radio frequency
PW = Pulse Width
PA = Pulse Amplitude
TOA= Time of Arrival as measured by the receiver clock
AOA= Angle of Arrival
Shaded areas indicate unused bit locations
P_O = Pulse Width Overflow
P_R = Pulse Amplitude Overrange

Figure 3 - Input Intercept Word Format

partitioned into two separate areas, each of which is addressed by frequency. The task of creating this address is functionally represented by the transform block of Figure 2. The break point for the feasibility model is programmable and may be any one of the lowest 16 pulse width values. The time increments represented by these values for the entire 6-bit pulse width field is shown in Table I.

Note that it is likely that several emitters would have F and PW values which could match the intercept value. All such emitters after the first are stored in an overflow area of the emitter file as shown in Figure 4. A link field in each emitter word is used to point to the location of each succeeding emitter tied to a given hash address. In this manner a chain of candidate emitters is constructed. Emitters are placed on the chain by the multiprocessor according to their PRI values *such that the emitters looked for most often* (i.e., have the highest PRF) appear at the heads of their respective chains.

Once the hash address has been developed, the search begins by accessing the emitter stored at this address and placing it in the data register as shown in Figure 5. The address development and memory access times for the feasibility model are approximately 153 and 450 n.sec respectively. Thus the emitter word arrives at the data register for comparison at approximately the same time as the second half of the intercept is clocked into the input register. Correlation then proceeds via comparisons on the fields

Table I
Pulse Width Bin Values

INPUT INTERCEPT ENCODED VALUE	EMITTER PULSE WIDTH		INPUT INTERCEPT ENCODED VALUE	EMITTER PULSE WIDTH	
	MIN µSEC	MAX µSEC		MIN µSEC	MAX µSEC
0	0.0	0.1	32	6.1	6.4
1	0.1	0.2	33	6.4	6.8
2	0.2	0.3	34	6.8	7.2
3	0.3	0.4	35	7.2	7.6
4	0.4	0.5	36	7.6	8.0
5	0.5	0.6	37	8.0	8.4
6	0.6	0.7	38	8.4	8.8
7	0.7	0.8	39	8.8	9.2
8	0.8	0.9	40	9.2	9.7
9	0.9	1.0	41	9.7	10.2
10	1.0	1.2	42	10.2	10.7
11	1.2	1.4	43	10.7	11.2
12	1.4	1.6	44	11.2	11.7
13	1.6	1.8	45	11.7	12.2
14	1.8	2.0	46	12.2	12.8
15	2.0	2.2	47	12.8	13.4
16	2.2	2.4	48	13.4	14.0
17	2.4	2.6	49	14.0	14.6
18	2.6	2.8	50	14.6	15.3
19	2.8	3.0	51	15.3	16.0
20	3.0	3.2	52	16.0	16.7
21	3.2	3.4	53	16.7	17.4
22	3.4	3.6	54	17.4	18.2
23	3.6	3.8	55	18.2	19.0
24	3.8	4.0	56	19.0	19.8
25	4.0	4.3	57	19.8	20.7
26	4.3	4.6	58	20.7	21.6
27	4.6	4.9	59	21.6	22.5
28	4.9	5.2	60	22.5	23.5
29	5.2	5.5	61	23.5	24.5
30	5.5	5.8	62	24.5	25.5
31	5.8	6.1	63	25.5	—

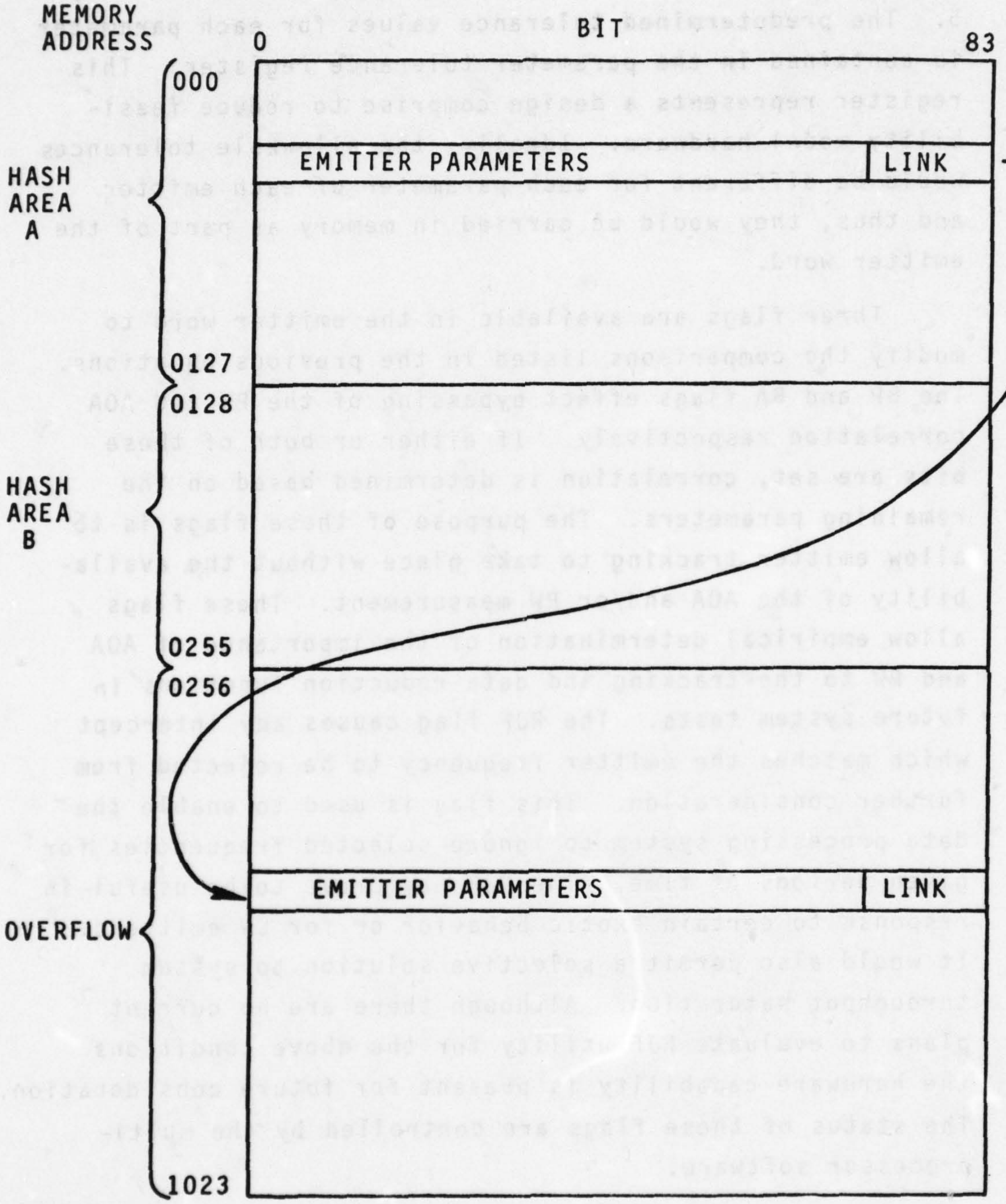


Figure 4 - Emitter File Memory Structure

shown in Figure 5 in accordance with equations 1 through 5. The predetermined tolerance values for each parameter is contained in the parameter tolerance register. This register represents a design comprise to reduce feasibility model hardware. Ideally, the allowable tolerances would be different for each parameter of each emitter and thus, they would be carried in memory as part of the emitter word.

Three flags are available in the emitter word to modify the comparisons listed in the previous equations. The BP and BA flags effect bypassing of the PW and AOA correlation respectively. If either or both of these bits are set, correlation is determined based on the remaining parameters. The purpose of these flags is to allow emitter tracking to take place without the availability of the AOA and/or PW measurement. These flags allow empirical determination of the importance of AOA and PW to the tracking and data reduction functions in future system tests. The RJF flag causes any intercept which matches the emitter frequency to be rejected from further consideration. This flag is used to enable the data processing system to ignore selected frequencies for given periods of time. The RJF may prove to be useful in response to certain exotic behavior or for CW emitters. It would also permit a selective solution to system throughput saturation. Although there are no current plans to evaluate RJF utility for the above conditions the hardware capability is present for future consideration. The status of these flags are controlled by the multiprocessor software.

Figure 6 depicts the logical flow for the TOA and frequency correlation. This figure shows the complexity that is typical of the two basic types of parameteric comparisons. The TOA and AOA are module 2^n comparisons

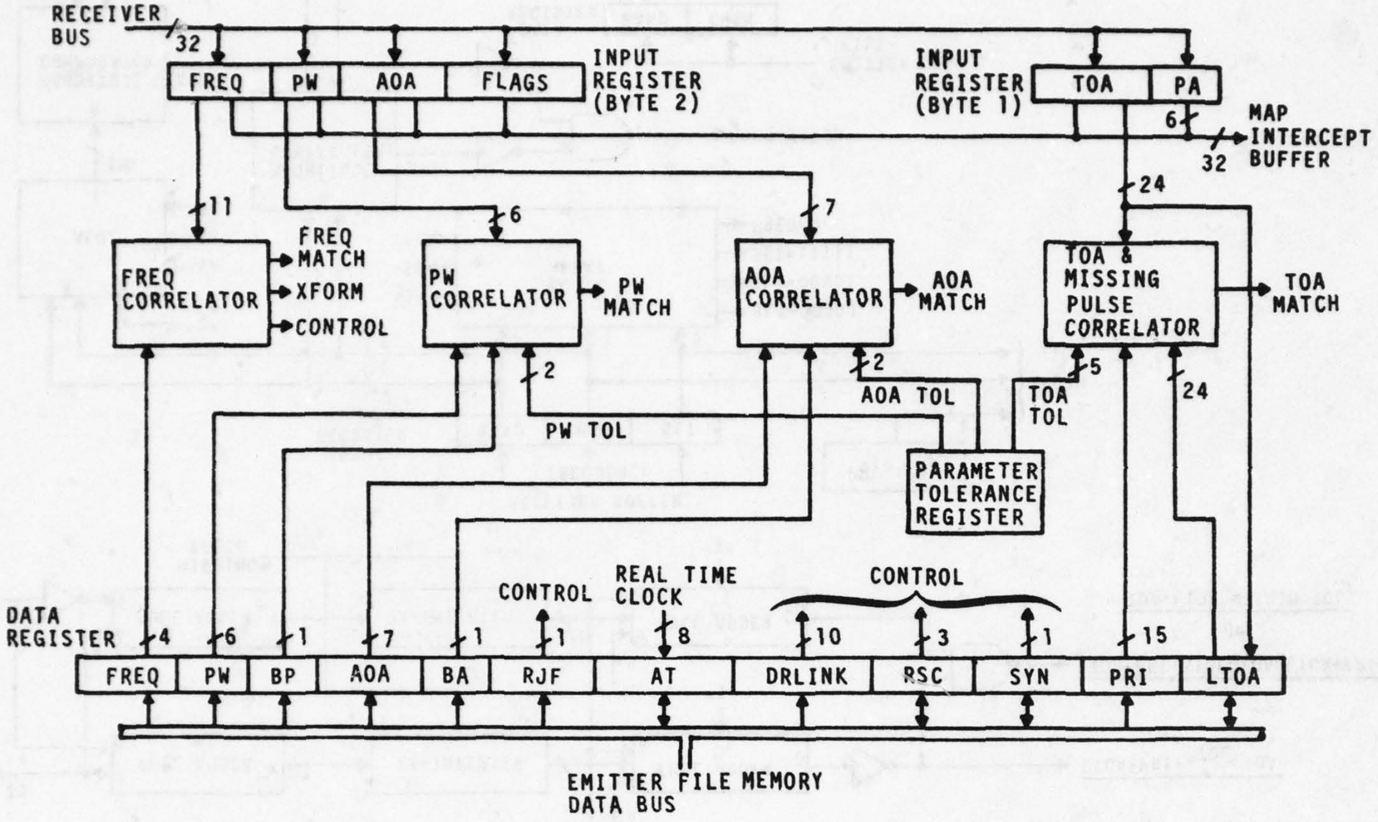


Figure 5 - Preprocessor Correlator

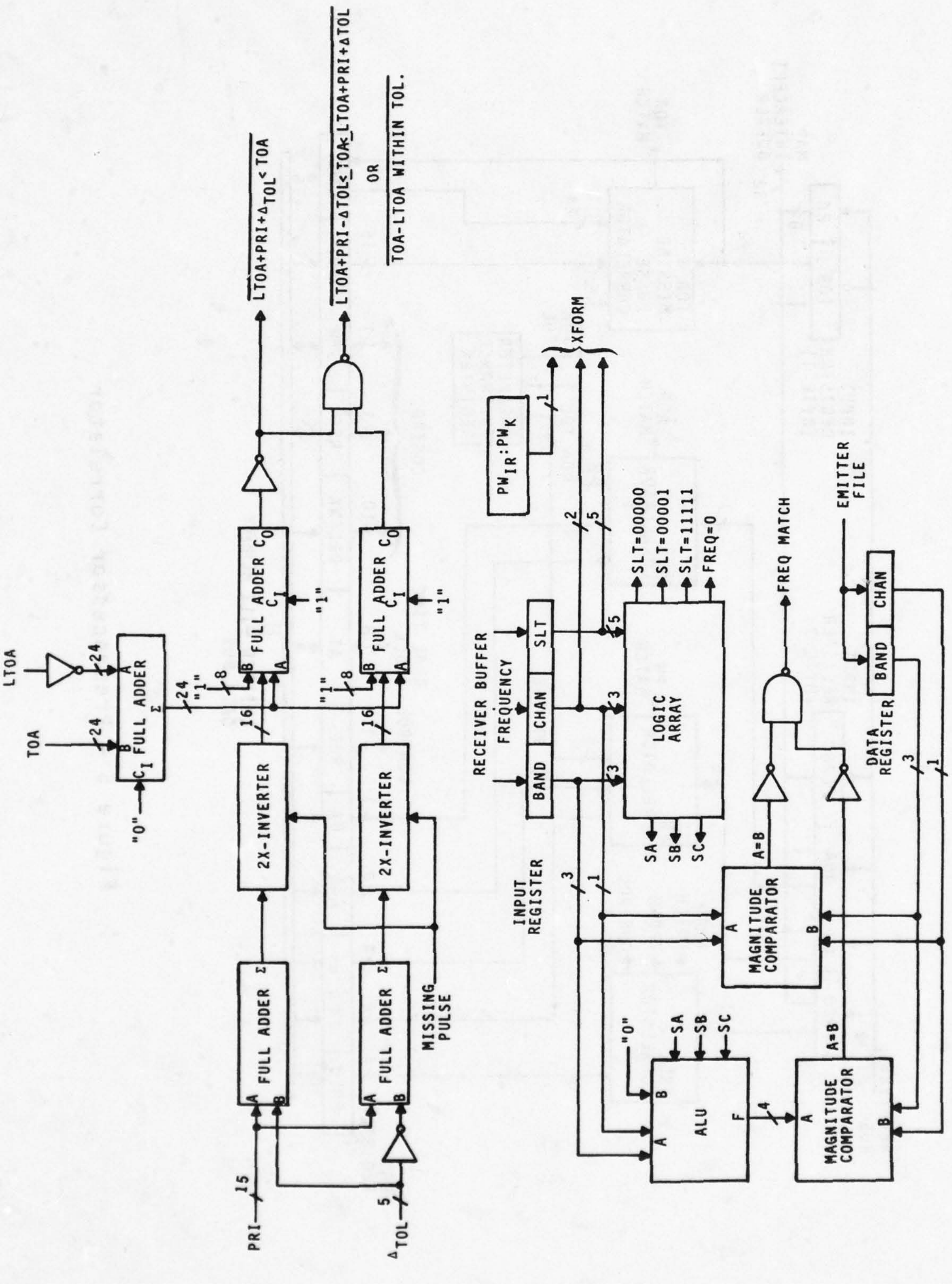


Figure 6 - TOA and Frequency Correlation Logic

(where n is the field length). That is the maximum and minimum binary numbers are considered adjacent values in the number scheme (i.e. end around comparisons). The F and PW comparisons, on the other hand treat the maximum and minimum binary values as opposite ends of the measurement spectrum.

If the parameter measurements of the intercept are within tolerance for the emitter, correlation is said to occur. The LTOA field of the emitter is upgraded to the TOA value of the intercept. Since the contributing emitter is found the intercept is dropped from further consideration.

If one or more parameter fails correlation, the search must continue. If the only failing parameter is TOA and it is greater than LTOA + 2 (PRI) the address of the emitter is placed on a stack for future reference before continuing the search. The search continues by accessing the location pointed to by the link field of the emitter in the data register to retrieve the next entry on the chain. The correlation process is then repeated. A link field of all one's indicates that the end of a chain has been reached.

When the end of the chain is reached without achieving correlation, the hash address on either side of the primary value is searched. If a correlating emitter is found on one of these chains that emitter's TOA value is updated and the pulse intercept is dropped from further consideration. If neither chain produces a correlation, a test is made to determine if the intercept PW value is sufficiently close to the break point to warrant searching of the other frequency hash area. Thus, up to a maximum of three more chains may have to be searched before correlation attempts are abandoned.

If none of the candidate chains produce a correlation it is likely that the intercept does not originate from any emitter currently in the file. An exception exists, however, for emitters on which the preprocessor has lost PRI track. The location of these emitters were placed on the stack during searching of the chain. Emitters (if any) referenced by the stack are then examined in a limited attempt to resync their LTOA fields. The resync count (RSC) field associated with each emitter contains the current number of attempts made to re-establish tracking of the emitters pulse train. If this field has already reached a maximum value no processing is performed on the emitter. Thus, only a limited number of attempts are made for any given emitter file word. If RCS is less than the maximum, the RCS field is incremented, the emitter LTOA value is replaced by the intercept TOA, and the age time field (AT) is updated to the preprocessor current real time clock value.

If the TOA assignment was correct for any stack emitter, it will again fully correlate with the next pulse intercept of the resumed train. Thus, the emitter word is back in step with its pulse train. When this happens the SYNC flag is set, the AT field is updated to current time and the RSC field is zeroed. Emitter words which do not become resynced to pulse trains are eventually age tested out of the file memory when their AT field falls too far behind real time. The purging of the emitter file is performed by one of the microprocessors of the multiprocessor subsystem.

The feasibility model stack implementation is a 16 word FIFO buffer. Thus, a maximum of 16 unsynced emitters may participate in the resyncing operation during a given intercept search procedure. The number of consecutive attempts to resync a given emitter is currently set to 4 although the RSC field is comprised of 3-bits which would allow the consecutive attempts to range from 0 to 8.

c Output Processing

Input intercepts which fail the correlation in the search operation and cannot be used in resync attempts are passed on to the multiprocessor subsystem for PRI establishment processing. The 64-bit intercept is transmitted as two 32-bit words over a single undirectional data bus. This bus is tied directly into two multiprocessor memory banks through buffered memory ports. Since the registers in the ports are dedicated, the preprocessor does not have to wait for a free multiprocessor memory cycle before initiating the transfer. The preprocessor loads one port with the first half of the intercept and then the second port with the remaining half. A common memory address is then sent over the data bus to both ports simultaneously. The transmission of the address also notifies the port logic of each memory bank that a memory request is pending.

A 2K word area in two, of the multiprocessor global, memory banks are reserved for the intercept passed on by the preprocessor. After 1024 intercepts are transfered, the preprocessor notifies the multiprocessor via interrupt that the buffer is half full. At this point the multiprocessor begins processing the first 1024 intercepts while the preprocessor fills the remainder of the buffer. When transfer of the second 1024 intercepts has been completed the two subsystems again swap buffer halves. An alternative mode of operation is also available where buffer swaps are a function of time rather than data rate. To effect this operation the multiprocessor loads a time interval count into the preprocessor which down counts this value to zero. At which time the preprocessor interrupts the multiprocessor and transfers a count of the number of pulse intercepts transferred during the interval. Buffer halves are then interchanged and the timer reset by the multiprocessor. The preprocessor currently initiates a buffer switch every 20 milliseconds.

In addition to keeping a count of the number of intercepts passed on to the multiprocessor per interval, the preprocessor also keeps a running total of the number of intercepts inputted to the subsystem. Both of these counts are available to the multiprocessor subsystem. Thus, the hardware capability exists to monitor input rate, output rate and extent of data reduction attained by the preprocessor. These functions are not currently supported by the multiprocessor software.

d System Control and Timing

The functional operations described in items a through c above are performed under microprogram control. Thus, a great deal of flexibility exist to modify the nature of the above search algorithm. The entire microprogramed algorithm is contained in eight programmable read only memories (PROM). Each PROM is organized as 256 words by 4 bits with a 50 n.sec. access time. A microprogram sequencer is used to drive the PROMS through this algorithm. The control hardware structure is shown in Figure 7 and a top level flowchart of the algorithm appears in Figure 8.

All decision elements from the data register and arithmetic sections form addresses into a second group of PROM's to comprise the branch capability of the control. The output of these PROM's define a new address (via the

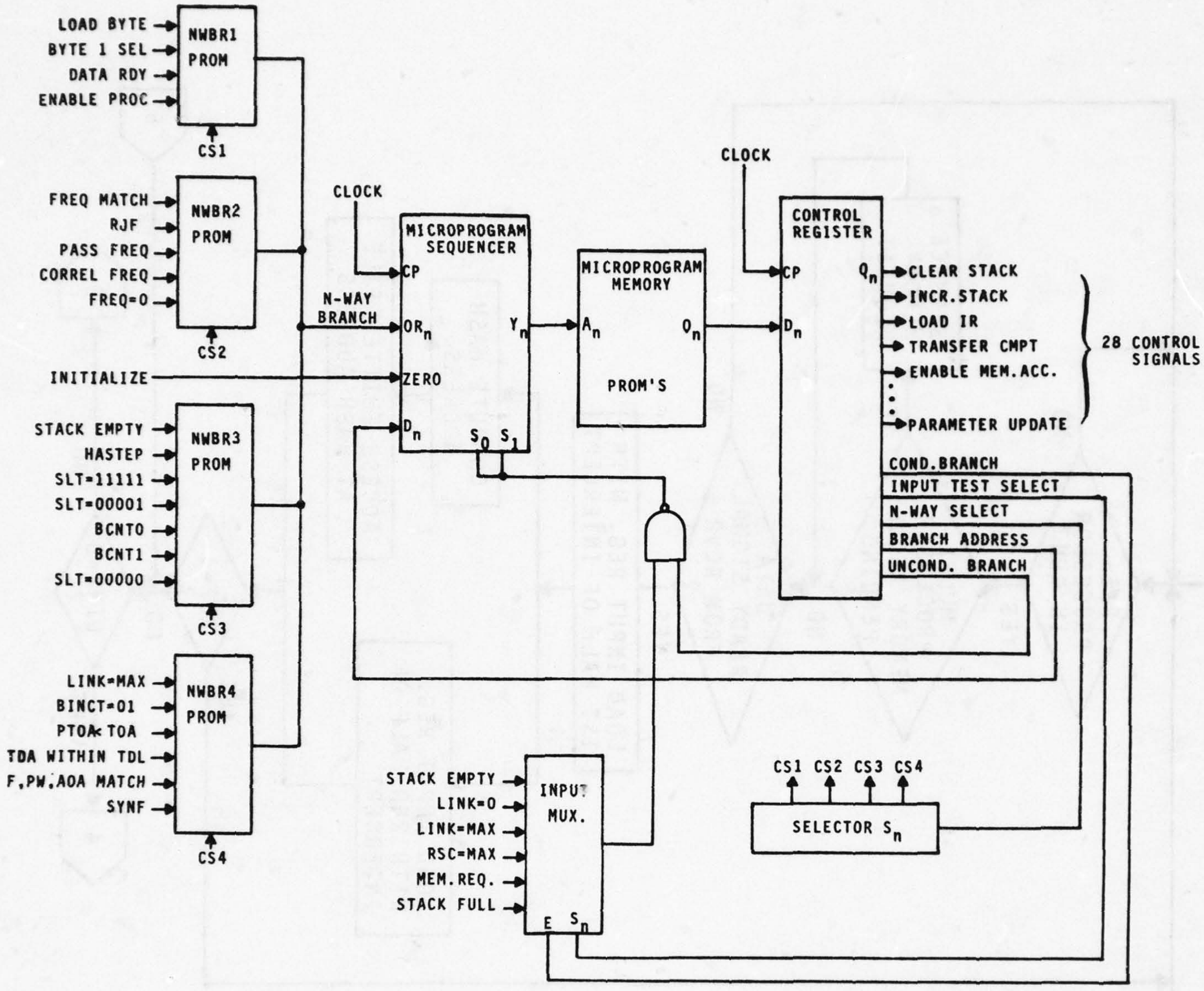


Figure 7 - Microprogram Control

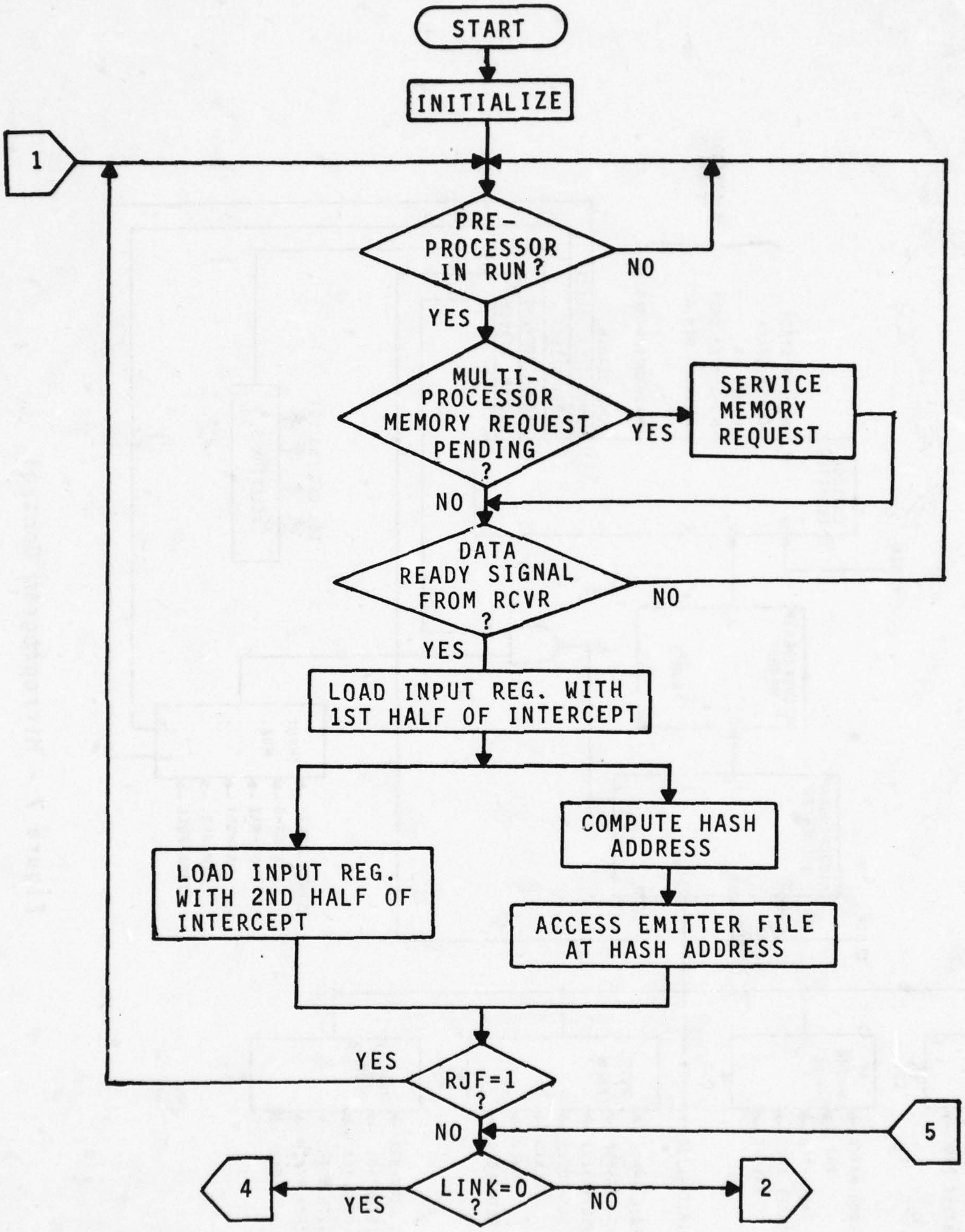

START
INITIALIZE
1
PRE-PROCESSOR IN RUN?
NO
YES
MULTI-PROCESSOR MEMORY REQUEST PENDING ?
YES
SERVICE MEMORY REQUEST
NO
DATA READY SIGNAL FROM RCVR ?
NO
YES
LOAD INPUT REG. WITH 1ST HALF OF INTERCEPT
COMPUTE HASH ADDRESS
LOAD INPUT REG. WITH 2ND HALF OF INTERCEPT
ACCESS EMITTER FILE AT HASH ADDRESS
YES
RJF=1 ?
NO
5
4
YES
LINK=0 ?
NO
2

Figure 8 - Top Level Preprocessor Search Algorithm Flowchart
Page 1 of 3

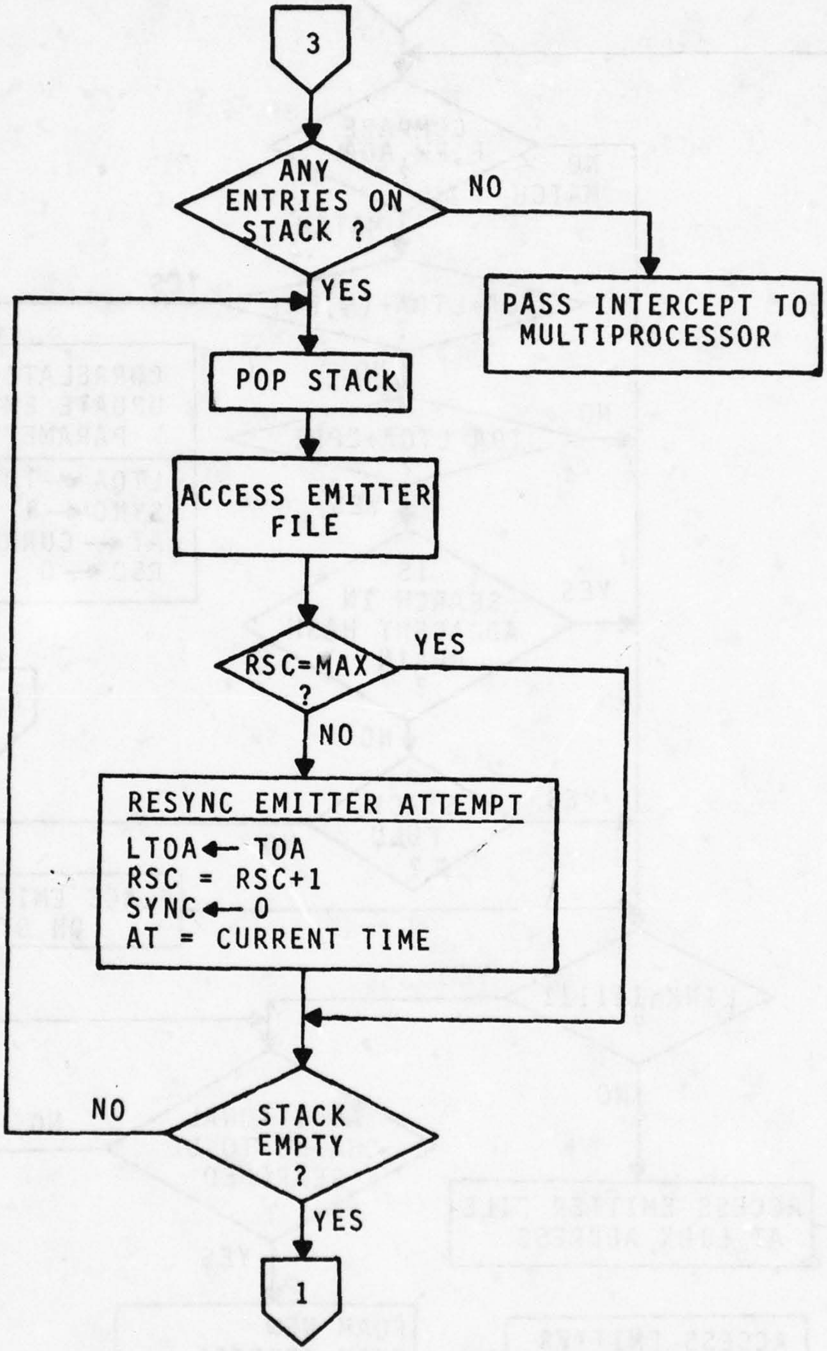


Figure 8 - Top Level Preprocessor Search Algorithm Flowchart
Page 2 of 3

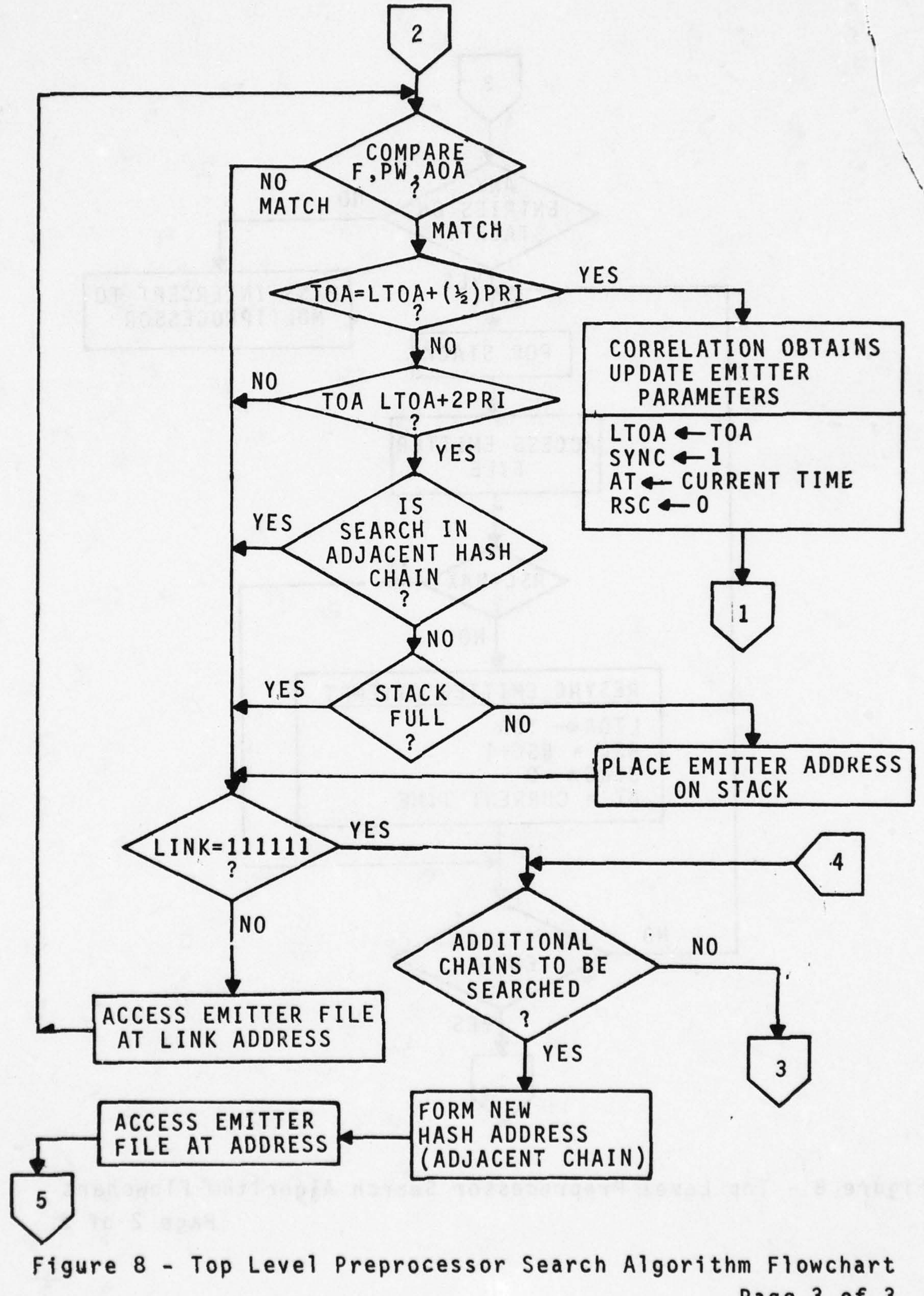

2
COMPARE F,PW,AOA ?
NO MATCH
MATCH
TOA=LTOA+(½)PRI ?
YES
NO
CORRELATION OBTAINS UPDATE EMITTER PARAMETERS
LTOA ← TOA
SYNC ← 1
AT ← CURRENT TIME
RSC ← 0
1
NO
TOA LTOA+2PRI ?
YES
YES
IS SEARCH IN ADJACENT HASH CHAIN ?
NO
YES
STACK FULL ?
NO
PLACE EMITTER ADDRESS ON STACK
LINK=111111 ?
YES
4
NO
ADDITIONAL CHAINS TO BE SEARCHED ?
NO
YES
ACCESS EMITTER FILE AT LINK ADDRESS
3
ACCESS EMITTER FILE AT ADDRESS
FORM NEW HASH ADDRESS (ADJACENT CHAIN)
5

Figure 8 - Top Level Preprocessor Search Algorithm Flowchart
Page 3 of 3

microprogram sequencer) for re-entering the microprogram code based on the input conditions (i.e., address to the branch PROM's). The advantage of this approach lies in the fact that in a single step 2^N conditions can be tested and a jump made to the appropriate section of microcode. Thus, many of the decision blocks of Figure 8 are executed in the same machine cycle of the preprocessor. All such groups of blocks are enclosed by dashed lines in the detailed flowchart in Appendix A. Each step through the microcode requires 153 nsec.

The time required to process a radar intercept for a number of typical input conditions for the feasibility model are shown in Table II. The first four conditions listed in this table cover cases where the input intercept successfully correlates with a stored emitter. The last three conditions represent cases where the intercept fails correlation and is passed on to the multiprocessor. The last column of the table shows equivalent preprocessor throughput rates if all input intercepts belong to a given type. In actual environments, the preprocess workload would consist of a mixture of intercept conditions with approximately 80 percent of the total intercepts composed of cases 1, 2 and 3. Thus, an average anticipated throughput rate would be approximately 200,000 pulses per second.

It should be noted that a substantial throughput improvement can be achieved by changing the fabrication approach from point-to-point wire wrap of logic panels to multilayer printed circuit boards. This switch would facilitate increasing the clock frequency from 6.5 MHz to 40 MHz and substitution of 50 nsec (access time) memory chips for the 450 nsec chips used for emitter file storage in the model. These changes would improve the throughput rate of the preprocessor by a factor of 6 or 7.

Table II

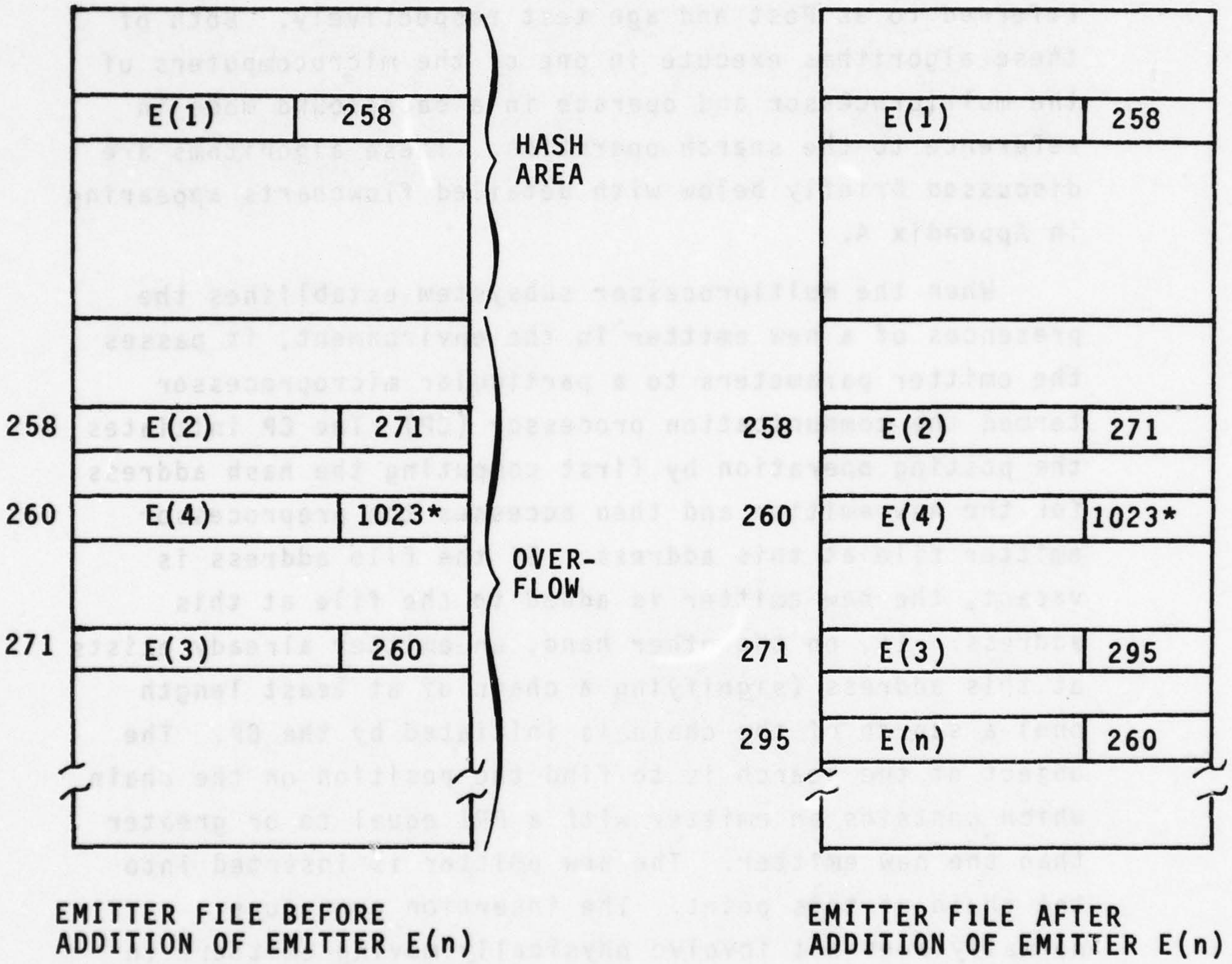
Preprocessor Processing Times

INPUT CONDITION	MACHINE CYCLES	EXECUTION TIME (μsec)	EQUIVALENT THROUGHPUT RATE (pulses per sec.)
1. INTERCEPT MATCHES FIRST EMITTER ON A CHAIN	13	1.98	502,800
2. INTERCEPT MATCHES SECOND EMITTER ON A CHAIN	19	2.90	344,000
3. INTERCEPT MATCHES THIRD EMITTER ON A CHAIN	25	3.82	261,400
4. INTERCEPT MATCHES FOURTH EMITTER ON A CHAIN	31	4.74	210,800
5. INTERCEPT DOES NOT CORRELATE (NO EMITTERS ON THE 3 CHAINS SEARCHED)	27	4.13	242,000
6. INTERCEPT DOES NOT CORRELATE - THREE CHAINS OF LENGTH 1 SEARCHED	29	4.44	225,300
7. INTERCEPT DOES NOT CORRELATE - THREE CHAINS OF LENGTH 2 SEARCHED	47	7.19	139,000

e Preprocessor Support Algorithms

In addition to the search algorithm which is implemented in preprocessor microcode, there are two other algorithms which are required to support preprocessor operation. These algorithms are responsible for composing and maintaining the preprocessor emitter file and are referred to as Post and age test respectively. Both of these algorithms execute in one of the microcomputers of the multiprocessor and operate in a background mode in reference to the search operation. These algorithms are discussed briefly below with detailed flowcharts appearing in Appendix A.

When the multiprocessor subsystem establishes the presences of a new emitter in the environment, it passes the emitter parameters to a particular microprocessor termed the communication processor (CP). The CP initiates the posting operation by first computing the hash address for the new emitter and then accesses the preprocessor emitter file at this address. If the file address is vacant, the new emitter is added to the file at this address. If, on the other hand, an emitter already exists at this address (signifying a chain of at least length one) a search of the chain is initiated by the CP. The object of the search is to find the position on the chain which contains an emitter with a PRI equal to or greater than the new emitter. The new emitter is inserted into the chain at this point. The insertion procedure normally does not involve physically moving emitters in the file, but only adjustments to several emitter links. This can best be shown with the aid of Figure 9 which depicts a chain before and after adding an emitter E(n) between the third and fourth emitters E(3),E(4) on a given



EMITTER FILE BEFORE
ADDITION OF EMITTER E(n)

EMITTER FILE AFTER
ADDITION OF EMITTER E(n)

*** The value of 1023 indicates the end of a chain.**

Figure 9 - Link Structure for Posting Emitters

chain. Here, E(n) is inserted into a vacant location in the overflow area of memory with a link value pointing to the location of emitter E(4). Then, the link field of emitter E(3) is changed to point to the location of the new emitter. Note-two memory accesses to the preprocessor file were required to insert the new emitter in its proper position on the chain and the physical address of all stored emitters remained unchanged. Since search takes precedence over the posting, the preprocessor may have processed numerous radar intercepts during the course of the post operation.

The CP also contains a bit map of the hash area of the emitter file. There is a one in the map for each hash address which has an emitter chain. This map is used during the age test operation to enable the CP to search only those areas of the preprocessor memory which contain emitter data. Age testing is done one emitter at a time and all the emitters on a given chain are tested before the CP moves to the next hash address. The test consists of comparing the emitter's age test field (AT) against the preprocessor current time clock. If the time difference is greater than a predetermined constant the emitter is removed from the chain. If the difference is less than the constant the search proceeds on to the next emitter. Removing an emitter consists of changing its predecessor link to point to its successor and adding the address of the age tested emitter to a table of empty addresses. The emitter does not have to be physically removed from the preprocessor memory.

The AT field for the feasibility model is 8-bits with a resolution of 16 milliseconds. The age test routine is entered once every 40 milliseconds based on an interrupt generated to CP by a timer in the preprocessor. Current CP software deletes any emitter from the file which has not correlated with an intercept for greater than 1 second.

3. FEASIBILITY MODEL PHYSICAL STRUCTURE

The feasibility model subsystem consists of 460 low-power Schottky TTL integrated circuits. The majority of these IC's are SSI and MSI packages with the remainder composed of LSI chips. The logic is housed on four Augat panels with point-to-point wire wrap used for the majority of the interconnections. The breakdown of the logic by function is shown in Table III. Total power consumption is under 40 watts.

A maintenance and test panel (M&T) is also provided with the preprocessor as a diagnostic tool and checkout aid. This panel consist of two 32-bit LED displays, a hexidecimal readout, keyboard input, and a number of control switches. A drawing of the M&T panel is shown in Figure 10, and a block diagram of its connection into the preprocessor logic shown in Figure 11. This panel allows an operator to single step through the preprocessor search algorithm and inspect the contents of the emitter file or the input register. The procedures for reading or writing a preprocessor register memory are shown in Table IV.

Because of the number of communication paths required between the multiprocessor and preprocessor for normal operation, it is also possible to perform substantial preprocessor testing from the multiprocessor. The preprocessor hardware which is accessible to the multiprocessor is shown in Figure 12. Functions which could be implemented via this channel include integrity testing, diagnostic testing, performance monitoring, and receiver simulation. Communication procedures between the two devices are detailed in Appendix B.

Table III

Preprocessor Logic Utilization

	AUGAT BOARD				
I.C.'s to support:	#1	#2	#3	#4	TOTAL
RCVR BUFFER INTRFC	12	0	15	0	27
M&T FUNCTIONS	6	10	41	25	82
INTERFACE TO MULTIPROCESSOR GLOBAL MEMORY	0	0	19	0	19
PREPROCESSOR FUNCTIONS	124	126	82	0	332
TOTAL PER BOARD	142	136	157	25	460

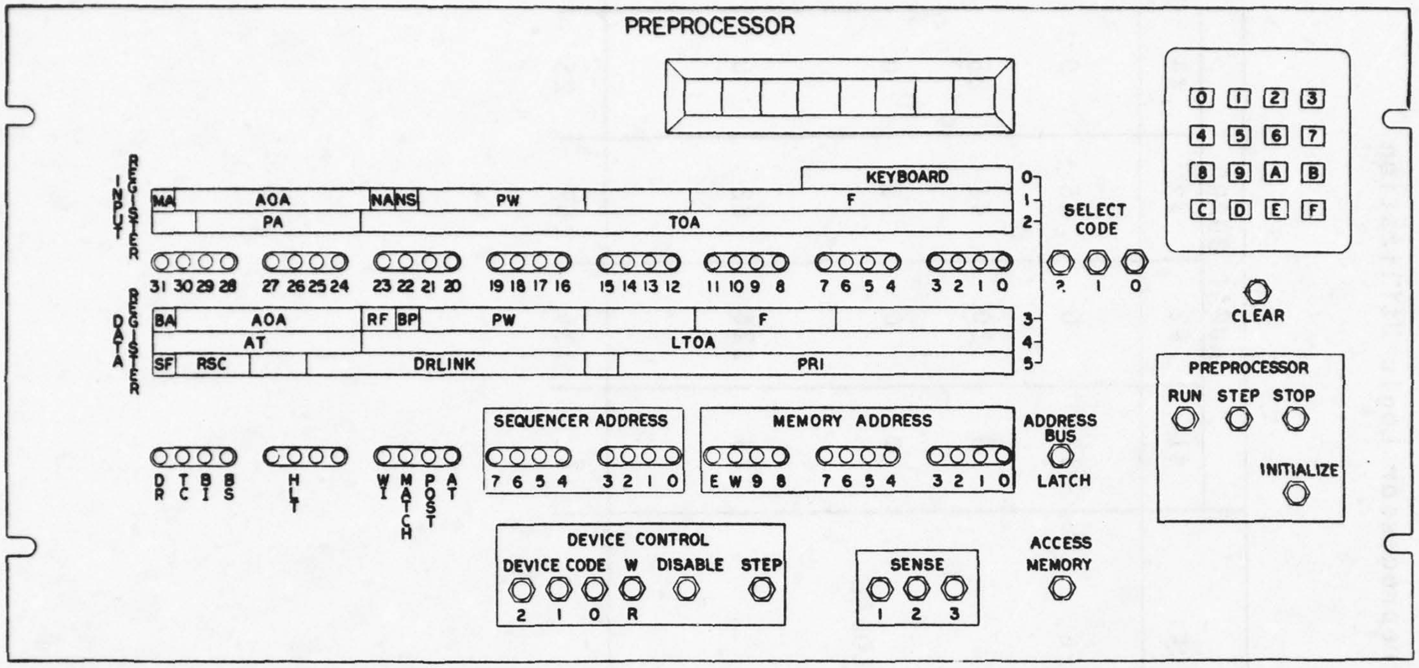


Figure 10 - Preprocessor M&T Front Panel

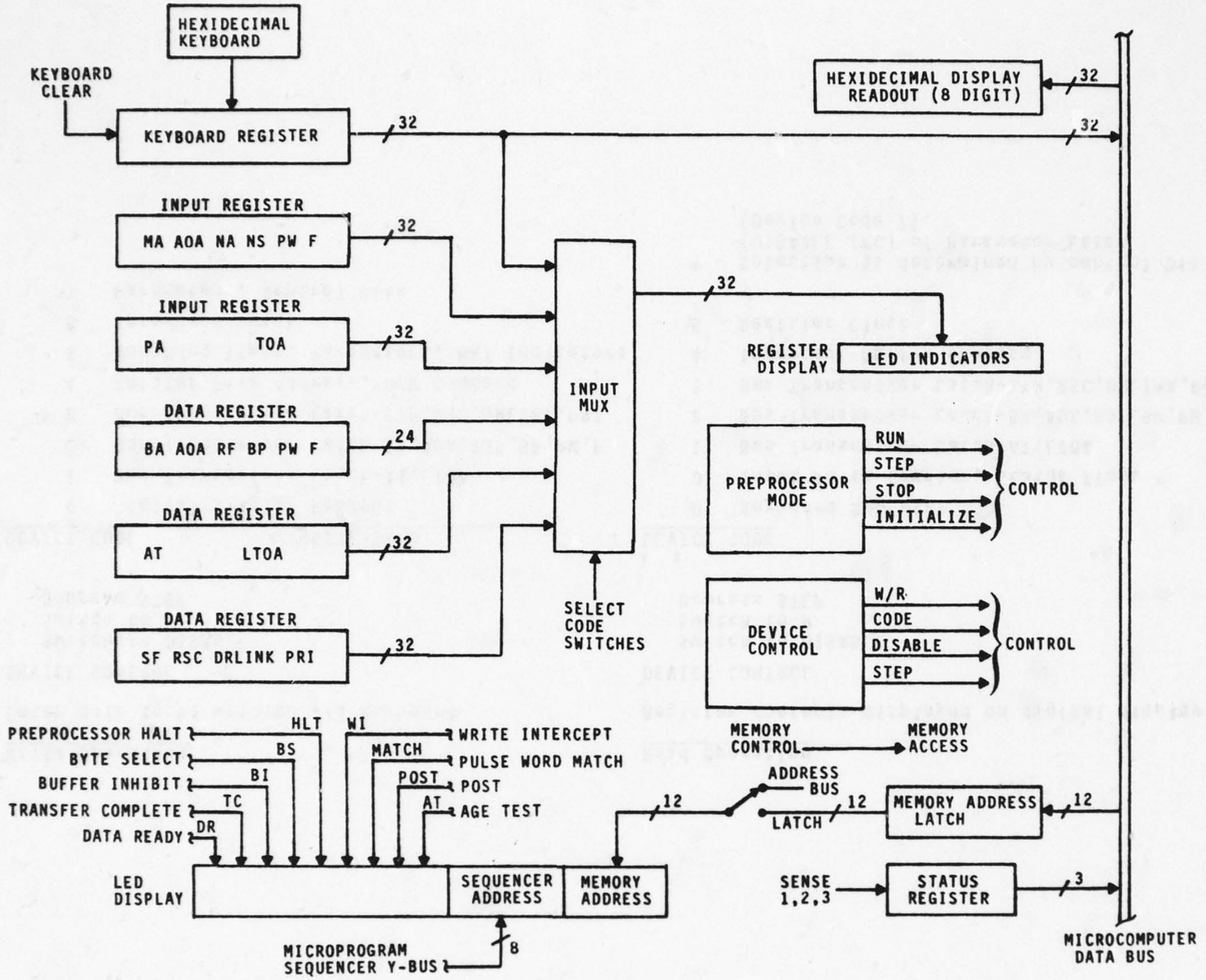


Figure 11 - Preprocessor M&T Functional Block Diagram

Table IV

Preprocessor M&T Panel Operations

Write Operation

Enter data to be written via keyboard

DEVICE CONTROL

Switch to DISABLE
Switch to W
Depress STEP

DEVICE CODE	WRITE INTO:
0	Digital Display Readout
1	Bus Transceiver Latch-AT,LTOA
2	Bus Transceiver Latch-BA,AOA,RJF,BP,PW,F
3	Bus Transceiver Latch-SYN,RSC,DRLINK,PRI
4	Emitter File Address, R/W Command
5	Watchdog Timer, Parameters, M&T Indicators
6	Intercept Latch
7	Parameters, Control Bits

Read Operation

Register contents displayed on digital display R.O.

DEVICE CONTROL

Switch to DISABLE
Switch to R
Depress STEP

DEVICE CODE	
0	Keyboard Register *
0	Input Pulse Counter, Status Flags *
1	Bus Transceiver Latch-AT,LTOA
2	Bus Transceiver Latch-BA,AOA,RJF,BP,PW,F
3	Bus Transceiver Latch-SYN,RSC,DRLINK,PRI
4	Intercept Buffer Address
5	Realtime Clock

* - Selection is determined by control bit (DISABLE IPC) of Parameter Latch (Device Code 7).

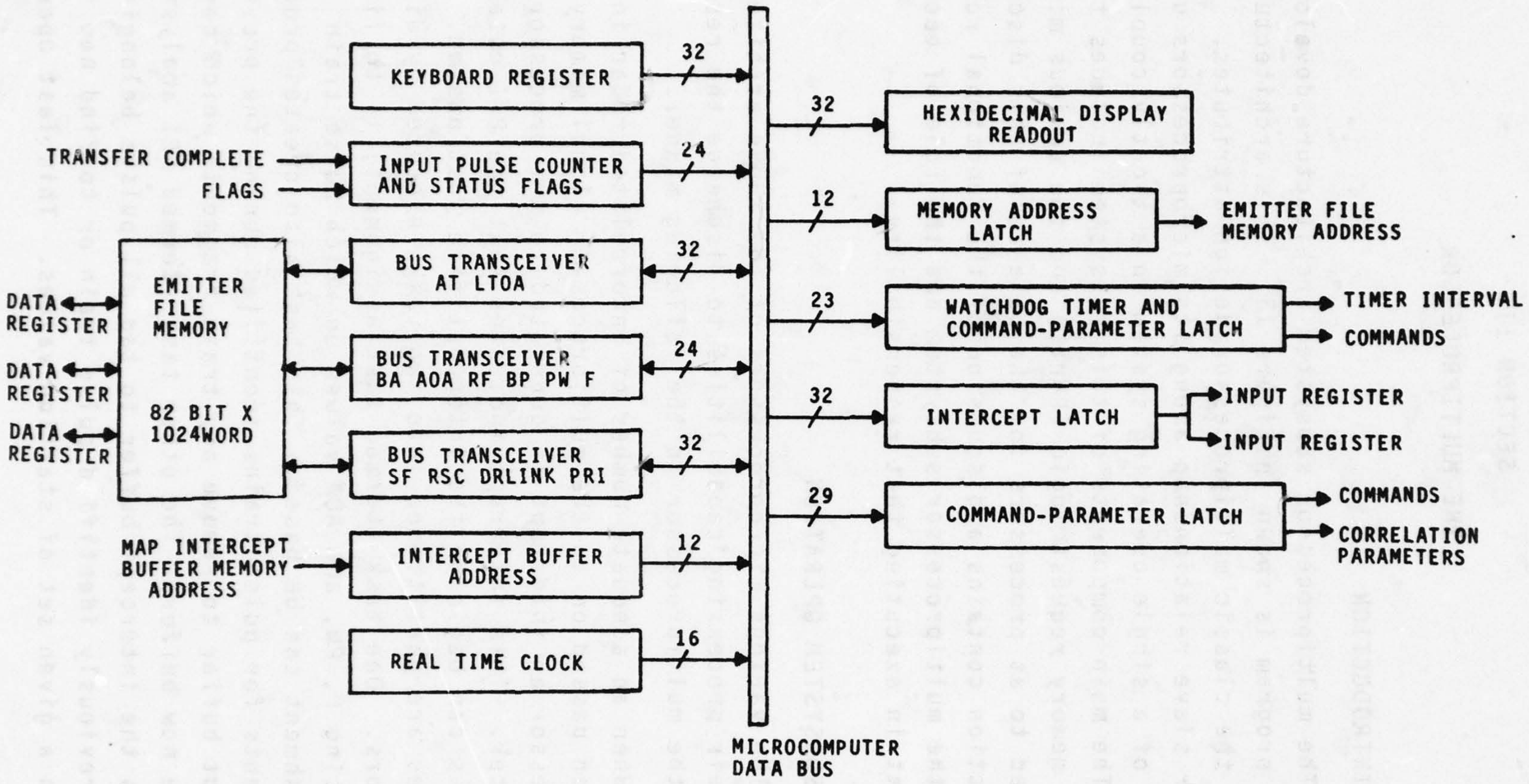


Figure 12 - Preprocessor/Microcomputer Communication Bus

SECTION III

THE MULTIPROCESSOR

1. INTRODUCTION

The multiprocessor subsystem architecture developed during the MAP program is shown in Figure 13. This architecture possesses many of the classic multiprocessor design attributes. It includes a master slave relationship among its microprocessors under the control of a single operating system in a tightly coupled structure. The main components of this subsystem includes the global memory, memory request logic (Ports) and the various microprocessor (referred to as processors for the balance of this discussion). This section contains a discussion of the functional responsibility of the multiprocessor subsystem and the role of each component in executing that responsibility.

2. SUBSYSTEM OPERATION

The various microprocessors of the above architecture pool their processing capabilities to discharge the responsibility of the multiprocessor in the following manner.

When an adequate number of uncorrelated radar intercepts have been passed on to the multiprocessor global memory by the preprocessor an interrupt is generated to the processor termed the master. This interrupt indicates that the PRI establishment processes can begin on the intercept file just passed. The processes are partitioned into two tasks executed by different processors. One task (termed task assignment) is to find sets of starting F, PW, and AOA values on which pulse train establishment can be based. This task also creates processing assignments for pulse trains identified during the previous intercept buffer to remove any train fragments which carry over into the new buffer. The other task (termed PRI analysis) searches the intercept buffer to tag all pulses belonging to a given previously identified pulse train or to find new trains based on a given set of starting values. This last operation

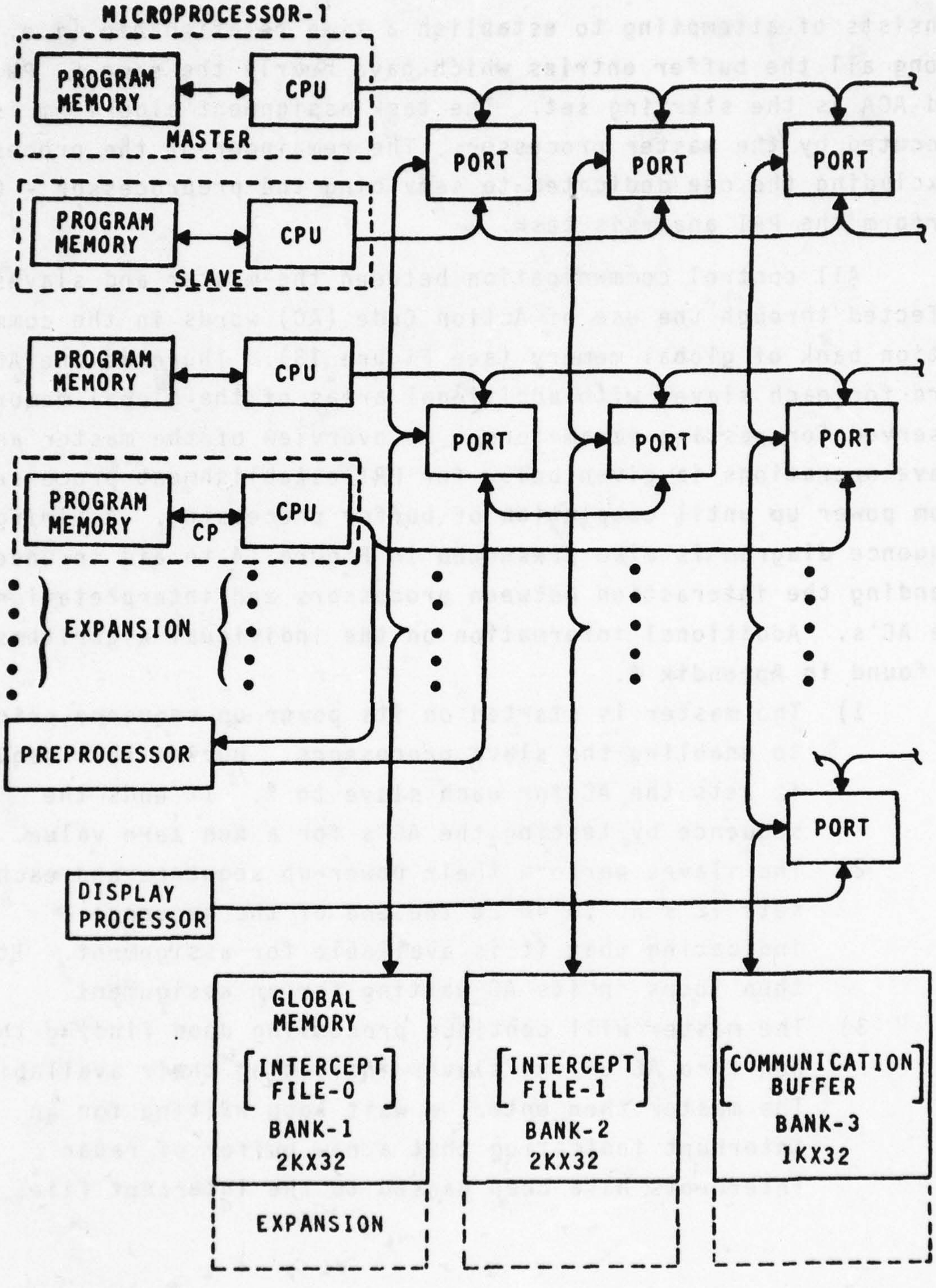


Figure 13 - Map Architecture

consists of attempting to establish a time relationship (i.e., PRI) among all the buffer entries which have nearly the same F, PW, and AOA as the starting set. The task assignment algorithm is executed by the master processor. The remainder of the processors (excluding the one dedicated to servicing the preprocessor - CP) perform the PRI analysis task.

All control communication between the master and slaves is effected through the use of Action Code (AC) words in the communication bank of global memory (see Figure 13). There is one AC word for each slave, with additional areas of the global memory reserved for passing parameters. An overview of the master and slave operations is given below for PRI establishment processes from power up until completion of buffer processing. A timing sequence diagram is also presented in Figure 14 to aid in understanding the interaction between processors and interpretation of the AC's. Additional information on the individual algorithms may be found in Appendix A.

1) The master is started on its power up sequence prior to enabling the slave processors. During this sequence it sets the AC for each slave to 0. It ends the sequence by testing the AC's for a non zero value.
2) The slaves perform their power-up sequence and each sets it's AC to 40 at the end of the sequence indicating that it is available for assignment. It then loops on its AC waiting for an assignment.
3) The master will continue processing upon finding the non-zero AC of the slaves indicating their availability. The master then enters a wait loop waiting for an interrupt indicating that a new buffer of radar intercepts have been passed to the intercept file.

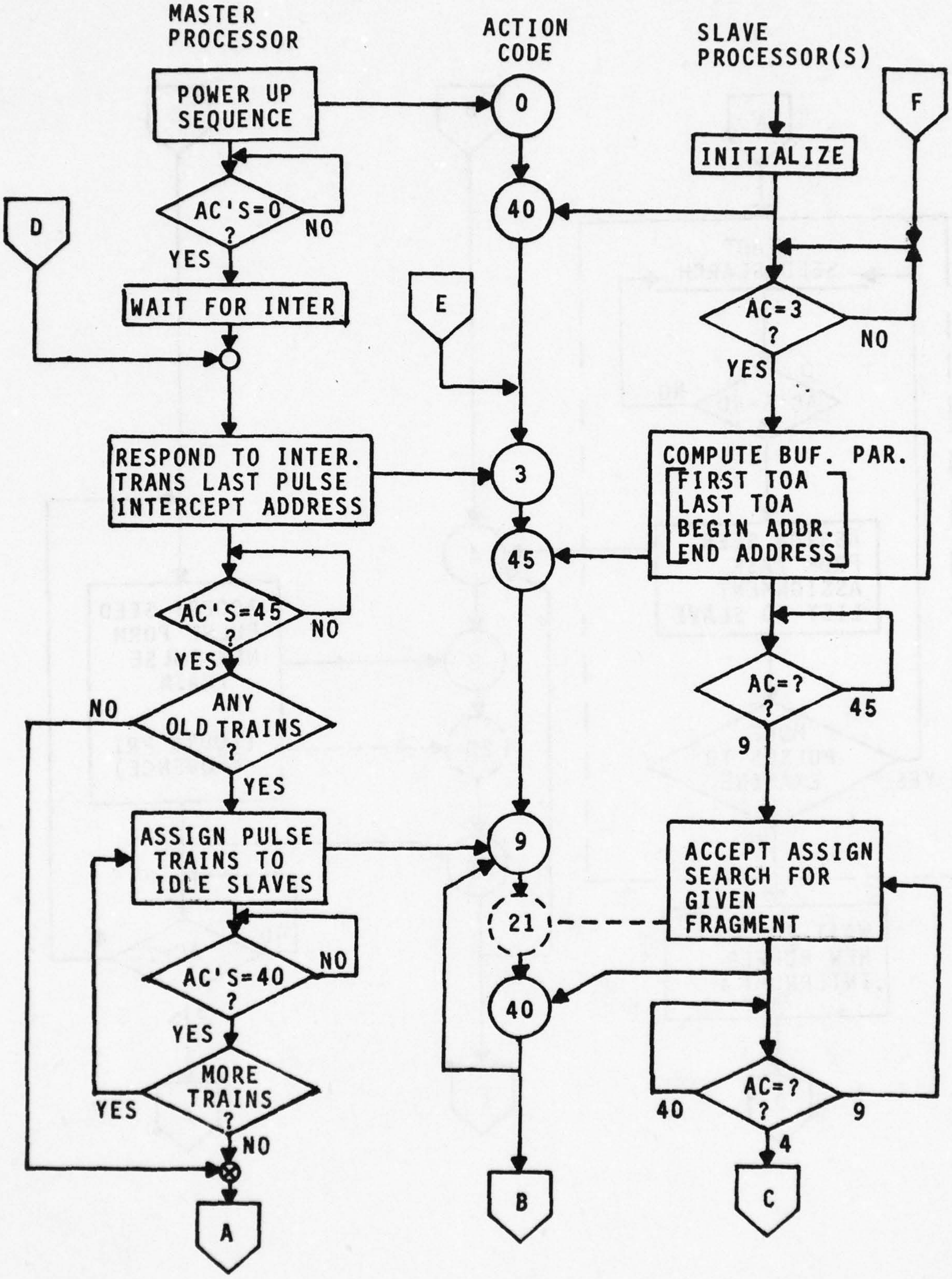


Figure 14 - Timing Sequence Flowchart (Sheet 1 of 2)

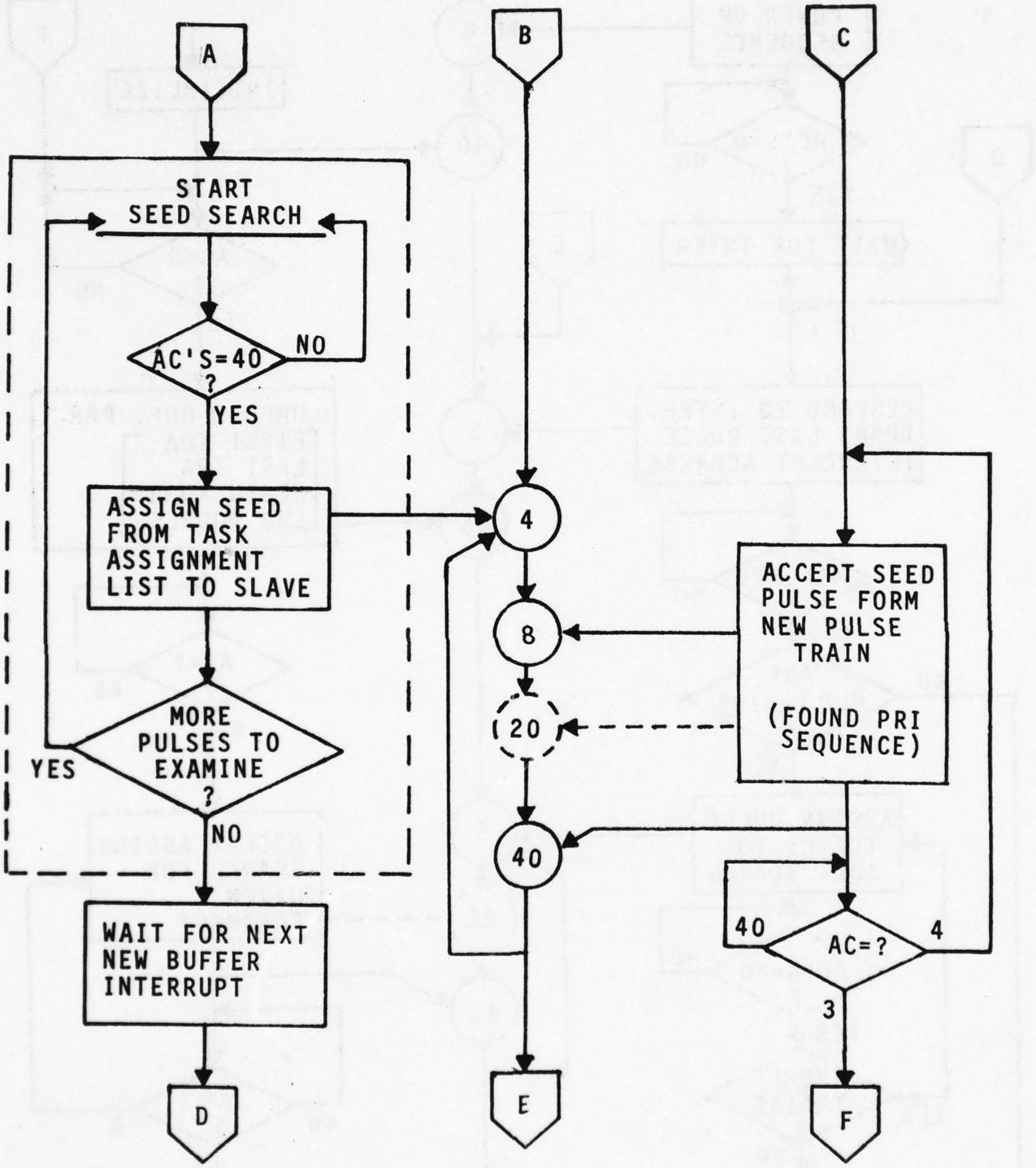

A
B
C
START SEED SEARCH
AC'S=40 ?
NO
YES
ASSIGN SEED FROM TASK ASSIGNMENT LIST TO SLAVE
MORE PULSES TO EXAMINE ?
YES
NO
WAIT FOR NEXT NEW BUFFER INTERRUPT
D
4
8
20
40
E
ACCEPT SEED PULSE FORM NEW PULSE TRAIN
(FOUND PRI SEQUENCE)
AC=?
40
4
3
F

Figure 14 - Timing Sequence Flowchart (Sheet 2 of 2)

4) Upon receiving the new buffer interrupt from the preprocessor, via the CP, the master begins new buffer processing. The master sets the AC to 3 for each available slave and stores the address of the last intercept in the new file in a message area for each slave.

5) The slave detects the 3 AC and initializes its internal parameters for analyzing the intercept file. The parameters include establishing the beginning buffer address, end of buffer address, first pulse and last pulse TOA's. Upon completion of this operation the slave sets his AC to 45. The slave then loops on the AC waiting for the next assignment.

6) When the master detects the 45 AC it assigns a pulse train found in the previous intercept buffer (if any) to the slave for train fragment removal. This is done by passing the train parameters to a message area and setting the slave AC to 9. The process is repeated until all available slaves receive assignments. If no trains were found in the previous buffer the master proceeds to step 9.

7) When a slave detects an AC of 9 it looks for pulses which belong to a previously identified emitter. This process consists of looking for intercepts which have the same F and AOA as the assigned train and also fit the PRI of the train. If three consecutive pulses are found which fit the PRI the slave changes its action code to 21 and tags all pulses which belong to the train. When this assignment is completed the slave returns its AC to 40. If 2 or less pulses are found they are not tagged and the action code is returned to 40 indicating slave availability. Upon completing the assignment the slave loops on the AC waiting for the next assignment.

8) The master detects the slaves availability (i.e., AC=40) and assigns the next train to the slave who is available. That is, steps 6 and 7 are repeated. This sequence is repeated until the master has no additional trains to assign. After passing the last assignment the master proceeds to step 9.

9) The master begins the search for starting points (seed pulses) for new train assignments. When a seed is found it is added to a master task assignment list. A seed is defined as a pulse which contains a F and AOA which fall outside of the limits of the values any slave is currently using (i.e., active assignments). Thus, when the master examines a seed candidate it also check to see which slaves are active and what F, AOA values they are using. When an idle slave is detected an assignment from the task list is given to the slave and its AC is set to 4. In handing off this assignment the master computes limits about F and AOA for searching purposes. This joint parametric space will not be available for seed assignment from this point on until the slave finishes the assignment.

10) When the slave detects an AC of 4, it initiates internal parameters for PRI establishment and changes its AC to 8. The seed parameters are retrieved from the message area for the given slave and a search is started in the intercept buffer to find another untagged pulse with similar F and AOA to form a tentative PRI. If such an intercept is found, the AC is changed to 10 by the slave. If two additional pulses are found, the existence of a train is confirmed and the AC is changed to 20. Parameters are averaged and the slave sets a report flag in global memory. The slave then finds all remaining pulses belonging to the train and tags them.

When finished, the AC is again returned to 40 by the slave indicating that is is available for assignment. If a slave cannot establish the presence of a train based on its seed values, the AC is also returned to 40. The slave then loops on the AC waiting for its next assignment. (Note: The master checks the report flags and stores each found train for future fragment analysis and passes the emitter to the preprocessor and display processor.

11) Steps 9 and 10 are repeated until the master has made two passes through the intercept buffer or a new buffer interrupt is received. At this point, the processing returns to step 4 and 4 through 11 are repeated.

The feasibility model of the multiprocessor contains four processors - one master - two slaves and one communication processor (which also executes the preprocessor support algorithms). The nature of this hardware and its interaction with other subsystem elements is covered below.

3. HARDWARE DESCRIPTION

a The Processor

The architecture of an individual processor of the multiprocessor subsystem is shown in Figure 15. The CPU section of this processor is configured around the 2901 bit slice CPU chip shown in Figure 16. This chip contains a scratch pad composed of 16 addressable registers. There are two address inputs (termed A and B) to the scratch pad so that two of its registers may be accessed concurrently. These two outputs along with the Q register and external input (Direct Data in) form the main inputs to the

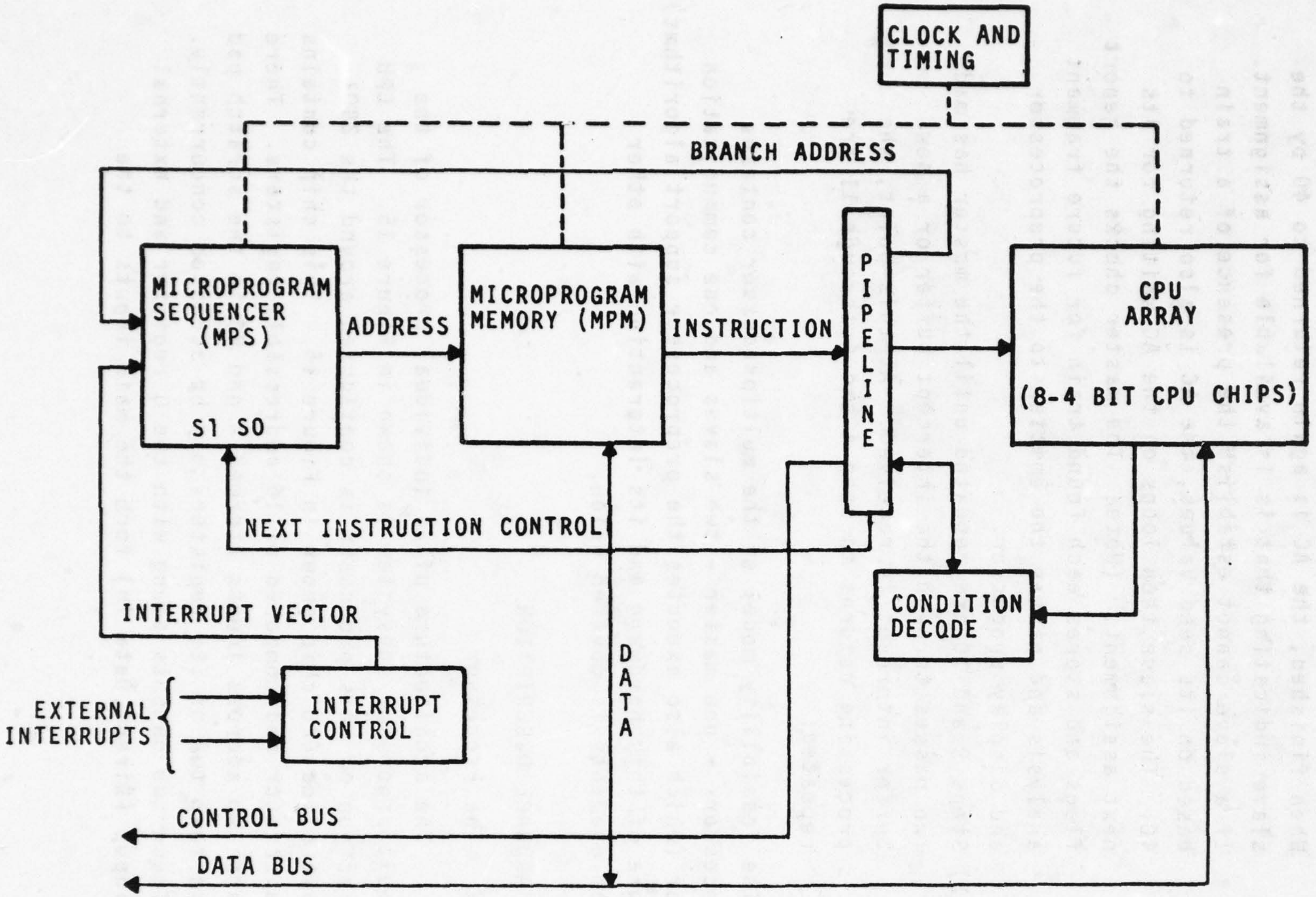


Figure 15 - Microprocessor Architecture

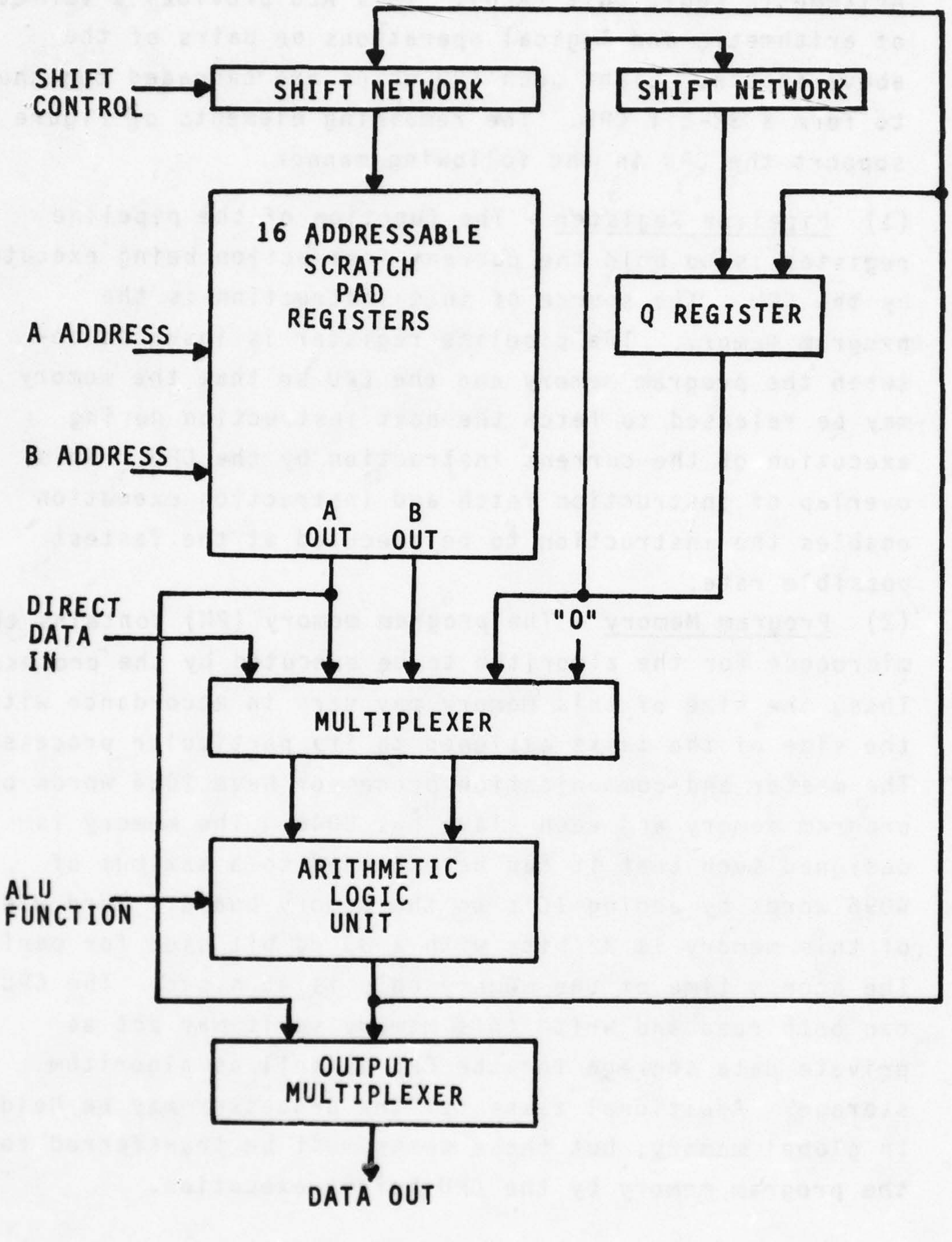


Figure 16 - The 2901 CPU Chip

Arithmetic Logic Unit (ALU). This ALU provides a variety of arithmetic and logical operations on pairs of the above inputs. Eight such CPU chips are cascaded together to form a 32-bit CPU. The remaining elements of Figure 15 support the CPU in the following manner.

(1) Pipeline Register - The function of the pipeline register is to hold the current instruction being executed by the CPU. The source of this instruction is the program memory. The pipeline register is inserted between the program memory and the CPU so that the memory may be released to fetch the next instruction during execution of the current instruction by the CPU. This overlap of instruction fetch and instruction execution enables the instruction to be executed at the fastest possible rate.

(2) Program Memory - The program memory (PM) contains the microcode for the algorithm to be executed by the processor. Thus, the size of this memory may vary in accordance with the size of the tasks assigned to its particular processor. The master and communication processor have 1024 words of program memory and each slave has 2048. The memory is designed such that it can be expanded to a maximum of 4096 words by adding IC's to the memory board. Word width of this memory is 32 bits with a 33'rd bit used for parity. The access time of the memory chip is 45 n.sec. The CPU can both read and write this memory so it may act as private data storage for the CPU as well as algorithm storage. Additional tasks for the processor may be held in global memory, but these tasks must be transferred to the program memory by the CPU before execution.

(3) Microprogram Sequencer - The function of the microprogram sequencer is address control of the program memory. The sequencer causes the program memory to sequence through its microcode in a proper order to effect execution of the data processing task. There are four possible sources of address for the sequencer. These are: 1) the program counter for execution of the instruction at the adjacent location to the present instruction, 2) the pipeline register for execution of jump commands, 3) a FIFO stack for return from subroutines, and 4) the interrupt circuitry for execution of interrupts. Additional details on the sequencer may be found in the first Interim Report.

(4) Clock and Timing - The clock circuitry is based on a Johnson counter whose subsequent length is a function of instruction type being executed. This allows each instruction type to be executed at the fastest rate possible determined by the combinational logic delays within the processor and line length to external devices.

(5) Condition Decode - The function of the decode logic is to facilitate conditional branching based on the value of any of the 32 bits of any CPU register or the carry out, overflow, and accumulator = 0 flags of the ALU.

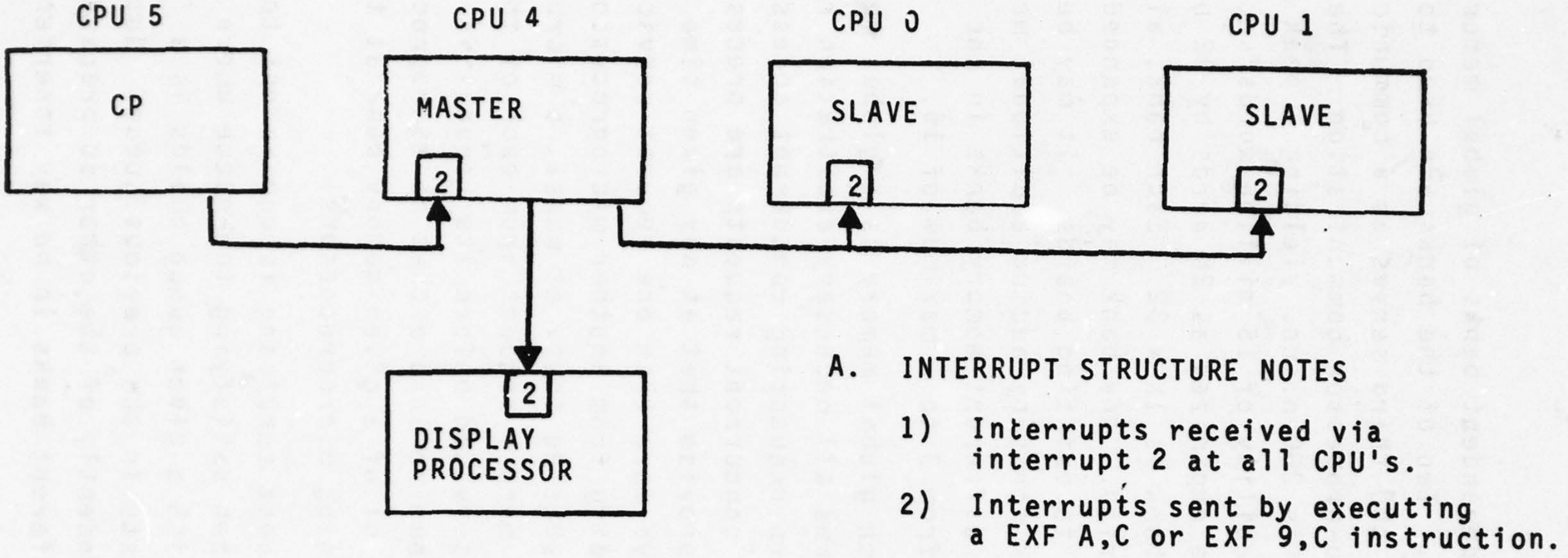
(6) Interrupt Control - The interrupt control allows the processor to respond to asychronous external stimulus without resorting to polling. It is also useful for implementation of response to certain internal fault conditions such as memory parity errors. There are a total of 8 levels of interrupt available to each CPU. Four of these levels are assigned to internal fault conditions. These interrupts are devoted to:

1) instruction bus parity error (i.e., program memory),
2) data bus parity error (i.e., from global memory or external device),
3) condition stack over/under flow, and
4) instruction cycle timeout (energized when an I/O instruction is not responded to in 8 usec.

The interrupt structure between processors is shown in Figure 17.

(7) Bus Structure - There are two busses associated with the microprocessor. One originates at the pipeline register (termed control bus) and the other connects to the CPU array (termed data bus). These two busses serve to tie the microprocessor to a common memory and other peripheral devices. The topology of this interconnection is discussed in the next item.

All of the above processor logic is partitioned onto two multilayer printed circuit boards which measure approximately 15.5 x 10 inches. One board contains the program memory and the other contains the CPU and remainder of the processor logic. The memory board consists of two outside signal layers with internal power and ground planes. The 1K version of this memory contains 40 integrated circuits and consumes approximately 15 watts. The CPU board is composed of 10 total layers which include two outside signal layers plus three internal signal layers on each side of the board and central power and ground planes. Adjacent signal layers contain lines which run predominantly perpendicular to each other to minimize noise coupling. This board contains 216 ICs and consumes approximately 20 watts.



A. INTERRUPT STRUCTURE NOTES

1) Interrupts received via interrupt 2 at all CPU's.

2) Interrupts sent by executing a EXF A,C or EXF 9,C instruction.

Figure 17 - Feasibility Model Interrupt Structure Between Processors

b Global Memory

There are three independent banks of global memory in the feasibility system. Two of the banks are used to house the intercept file and the third serves as a communication buffer for processor-to-processor communication. The effective memory cycle is 200 n.sec. yielding a peak memory subsystem transfer capability of 15 million words/second. The intercept banks are organized as 2K words by 32 bits and the communication bank is 1K x 32. Each bank, also contains one bit of parity. Any bank may be expanded to 4K words by adding ICs to existing boards. It may be further expanded to 64K words by adding additional memory boards. The number of different memory banks in the system may also be increased from 3 to a maximum of 16.

Associated with each global memory is a global memory controller which performs all necessary arbitration resulting from multiple processors requesting concurrent access to the same memory bank. All concurrent requests are processed sequentially with the proviso that at any given time no microprocessor will have more than one request serviced while a request is pending from another microprocessor. This is equivalent to stating that, at worst, a microprocessor may have to wait for a memory request from each of the other microprocessors to be serviced before its request is processed. This was done to insure that no microprocessor may gain complete control of a given memory bank at the exclusion of the remaining microprocessors.

This method of request servicing is equivalent to queueing them up and then satisfying the queue where the length of time over which a given queue builds is a function of the number of requests in the previous queue. Note each bank works independently of the other so processors making requests to different banks in no way interfer with each other.

c Port Logic

The purpose of the port logic is to provide a communication path between a device and a global memory bank. All processors of Figure 13 communicate with the global memory through a port. Each port contains a holding register and associated control to recognize that a request is for its memory bank. All requests (either read or write) are held in the port register until serviced by the global memory. Each port printed circuit board contains two ports which can be used to connect two separate devices to a given global memory bank. A total of eight ports can be supported by any given global memory bank.

For the feasibility model the master and both slaves has access to all three memory banks. The preprocessor has access through ports to only the two intercept file banks and the display processor to only the communication bank. There is one unused port in this system with the capability of adding additional port cards at a later date.

4. PROGRAMMING

The individual processors of the multiprocessor subsystem may be programmed at two levels. The most tedious of the two is the machine level where the algorithms are translated into the binary machine code by the programmer. The other level of programming supported by the system is the assembly language level which allows the programmer to work with mnemonics and labels rather than binary code. A brief description of the assembly language appears in section IV.

The set of operational instructions supported at the machine level are described in the following pages along with representative instruction execution times. This machine level instruction set encompassed all bit patterns of the 32-bit instruction word that is supported by the processor hardware.

OPCODE 0 Register/Register Operations

Description:

Offers control of ALU, 16 GP registers, Q register, and shift network. Any two of the 16 GP registers may be specified via fields A&B of the instruction format. Two source operands to the ALU, the ALU function, and destination of the ALU output are specified by the SORC, FUNC, DEST fields respectively. Field CN is the carry input to the ALU. Shifting is enabled by bit 25; left/right shift is controlled by bit 24. Shift mode is controlled by shift select (SFT SEL) field.

To accomodate an "add & shift" algorithm for hardware multiplication, bit 27 of the SORC field may be driven by the LSB of the Q register. The multiplier, multiplicand, and partial product reside in the Q, A, & B registers respectively. If bits 28 & 26 are 0 & 1 respectively, the source (SORC) operands will be A & B (add multiplicand to partial product) or 0 & B (add nothing to partial product) dependent upon the LSB of the Q register (Q_0). With each cycle, the Q & B registers are shifted one bit towards the LSB. If the just completed LSB of the product is deposited in the MSB of Q each cycle, the Q register fills with the least significant half of the product. With multiply mode flip flop set, bit 27 of the SORC is driven by $\overline{Q}_0$ (bit 27 in the instruction register is irrelevant); with the multiply mode flip flop reset, bit 27 of the SORC is driven from the instruction register.

OPCODE 0 REGISTER/REGISTER OPERATIONS

31 30 29	28 27 26	25 24 23	22 21 20	19 18 17 16	15 14 13 12	11	10 9	8	7	6 5 4 3 2 1 0
0 0 0	SORC	DEST	FUNC	B	A	CN	SFT SEL	X	NOTE 5)	X X X X X X X

DEST 25 24 23	Load B Register	Load Q Register	Load Status Register	OPERATION DESCRIPTION
0 0 0	NC	NC	Note 5)	NOP or Compare
0 0 1	NC	F	F/CC	Load Q
0 1 0	F	NC	F/CC	Load B
0 1 1	F	NC	A/CC	Load B; Load Status with A
1 0 0	F/2	NC	F/CC	Shift B towards LSB
1 0 1	F/2	Q/2	F/CC	Shift B & Q towards LSB
1 1 0	2F	NC	F/CC	Shift B towards MSB
1 1 1	2F	2Q	F/CC	Shift B & Q towards MSB

24	SFT SEL 10 9	Shift DIRECTION	Load B_{31}	Load B_0	Load Q_{31}	Load Q_0	TYPE OF SHIFT
0	0 0		0	F_1	0	Q_1	Logical Zero
0	0 1	Down	1	F_1	1	Q_1	Logical One
0	1 0	(Towards	F_0	F_1	Q_0	Q_1	Single Precision Rotate
0	1 1	LSB)	$N \oplus V$	F_1	F_0	Q_1	Double Precision Arithmetic
1	0 0		F_{30}	0	Q_{30}	0	Logical Zero
1	0 1	Up	F_{30}	1	Q_{30}	1	Logical One
1	1 1	(Towards	F_{30}	F_{31}	Q_{30}	Q_{31}	Single Precision Rotate
1	1 1	MSB)	F_{30}	Q_{31}	Q_{30}	0	Double Precision Arithmetic

SORC 28 27 26	ALU SOURCE OPERANDS R	 S
0 0 0	A	Q
0 0 1	A	B
0 1 0	0	Q
0 1 1	0	B
1 0 0	0	A
1 0 1	Unpredictable	
1 1 0	Unpredictable	
1 1 1	Unpredictable	

CN 11	FUNC 22 21 20	Function	OPERATION DESCRIPTION
0	0 0 0	R+S	Add
0	0 0 1	S-R-1	Subtract (1's comp)
0	0 1 0	R-S-1	Subtract (1's comp)
0	0 1 1	$R \vee S$	Logical inclusive or
0	1 0 0	$R \wedge S$	Logical AND
0	1 0 1	$\overline{R} \wedge S$	Logical complement AND
0	1 1 0	$R \oplus S$	Logical exclusive OR
0	1 1 1	$\overline{R \oplus S}$	Logical exclusive NOR
1	0 0 0	R+S+1	ADD + 1
1	0 0 1	S-R	SUBTRACT (2's comp)
1	0 1 0	R-S	SUBTRACT (2's comp)
1	0 1 1	$R \vee S$	
1	1 0 0	$R \wedge S$	CN Bit
1	1 0 1	$\overline{R} \wedge S$	Irrelevant to
1	1 1 0	$R \oplus S$	Logical Functions
1	1 1 1	$\overline{R \oplus S}$	

Notes:

1) NC indicates NO CHANGE.
2) F indicates DATA OUTPUT OF ALU.
3) CC indicates ALU CONDITION CODES - N,V,C,Z.
4) X indicates DON'T CARE
5) NC TO STATUS REGISTER IF BIT 7 EQUAL ZERO (DEST=0);
   LOAD STATUS REGISTER WITH F/CC IF BIT 7 SET (DEST=0).

Instruction Timing		
Bit 25=0 AND multiply mode FF Reset		250ns
Bit 25=1 OR multiply mode FF Set		325ns

OPCODE 1 Input/Output Operations

Description:

Provides for CPU communication with other devices interfaced with the CPU bus. The device is specified by the DEVICE ADDRESS field; data flow direction is specified by the SORC field. CPU operation is controlled by the SORC, DEST, FUNC, B, A, and CN fields. Any two of the 16 GP registers may be specified via fields A & B of the instruction format. Two source operands to the ALU, the ALU function, and destination of the ALU output are specified by the SORC, FUNC, DEST fields respectively. Field CN is the carry input to the ALU.

If the external device (specified by bits 0-5) does not acknowledge a CPU I/O instruction cycle, the CPU will repeat the request every 250ns until 8us has elapsed, at which point the CPU issues interrupt 6 and continues to the next instruction in sequence.

Bus parity is checked at the end of an input instruction cycle. If an error is found, the CPU issues interrupt 5.

OPCODE 1 provides a capability to present the contents of a GP register to the CPU bus and increment/decrement the register in the same instruction cycle. Choose SORC=4 and DEST=3; increment/decrement is controlled via FUNC and CN. Register A is presented to the CPU bus while A±1 is loaded into B.

OPCODE 1 INPUT/OUTPUT OPERATIONS

31 30 29	28 27 26	25 24 23	22 21 20	19 18 17 16	15 14 13 12	11	10 9 8 7 6	5 4 3 2 1 0
0 0 1	SORC	DEST	FUNC	B	A	C N	X X X X X	DEVICE ADDRESS

DEST 25 24 23	Load B Register	Load Q Register	Load Status Register	SOURCE TO BUS SORC= 0-4	 SORC= 5-7	OPERATION DESCRIPTION
0 0 0	NC	NC	F/CC	F	EXTERNAL DEVICE	Load Status Register Only
0 0 1	NC	F	F/CC	F		Load Q
0 1 0	F	NC	F/CC	F		Load B
0 1 1	F	NC	A/CC	A		Load B; Load Status with A
1 0 0	CNP	NC	F/CC	F		NOT RECOMMENDED
1 0 1	CNP	CNP	F/CC	F		NOT RECOMMENDED
1 1 0	CNP	NC	F/CC	F		NOT RECOMMENDED
1 1 1	CNP	CNP	F/CC	F		NOT RECOMMENDED

SORC 28 27 26	ALU SOURCE OPERANDS R	 S	DATA FLOW DIRECTION
0 0 0	A	Q	OUTPUT
0 0 1	A	B	OUTPUT
0 1 0	0	Q	OUTPUT
0 1 1	0	B	OUTPUT
1 0 0	0	A	OUTPUT
1 0 1	BUS	A	INPUT
1 1 0	BUS	Q	INPUT
1 1 1	BUS	0	INPUT

CN 11	FUNC 22	 21	 20	FUNCTION	OPERATION DESCRIPTION
0	0	0	0	R+S	Add
0	0	0	1	S-R-1	Subtract (1's comp)
0	0	1	0	R-S-1	Subtract (1's comp)
0	0	1	1	$R \vee S$	Logical inclusive OR
0	1	0	0	$R \wedge S$	Logical AND
0	1	0	1	$\overline{R} \wedge S$	Logical complement AND
0	1	1	0	$R \oplus S$	Logical Exclusive OR
0	1	1	1	$\overline{R \oplus S}$	Logical Exclusive NOR
1	0	0	0	R+S+1	ADD + 1
1	0	0	1	S-R	SUBTRACT (2's comp)
1	0	1	0	R-S	SUBTRACT (2's comp)
1	0	1	1	$R \vee S$	CN BIT IRRELEVANT TO LOGICAL FUNCTIONS
1	1	0	0	$R \wedge S$	
1	1	0	1	$\overline{R} \wedge S$	
1	1	1	0	$R \oplus S$	
1	1	1	1	$\overline{R \oplus S}$	

Notes:

1) NC indicates NO CHANGE.
2) F indicates DATA OUTPUT OF ALU.
3) CC indicates ALU CONDITION CODES-N,V,C,Z.
4) CNP indicates UNPREDICTABLE.
5) X indicates DON'T CARE.

Instruction Timing	
OUTPUT (SORC=0-4)	350ns
INPUT (SORC=5-7)	400ns

OPCODE 2 Register/Immediate Operations

Description:

Allows the Q Register or any of the sixteen (16) GP Registers to be loaded without a data fetch. Sixteen (16) leading zeros are appended to the IMMEDIATE VALUE in the instruction format to provide a 32-bit operand to the ALU. The second operand to the ALU, the ALU function, and destination of the ALU output are specified by the SORC, FUNC & DEST fields respectively. Any one of the 16 GP registers may be specified by field B of the instruction format. If field H (bit 28) is set, the 32-bit immediate operand is shifted 16 bits end-around before being presented to the ALU.

OPCODE 2 REGISTER/IMMEDIATE OPERATIONS

31 30 29	28	27 26	25 24 23	22 21 20	19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 1 0	H	SORC	DEST	FUNC	B	IMMEDIATE VALUE (I)

DEST 25 24 23	Load B Register	Load Q Register	Load Status Register	OPERATION DESCRIPTION
0 0 0	NC	NC	F/CC	Load Status Register only
0 0 1	NC	F	F/CC	Load Q
0 1 0	F	NC	F/CC	Load B
0 1 1	F	NC	B/CC	Load B; Load Status with B
1 0 0	CNP	NC	F/CC	NOT RECOMMENDED
1 0 1	CNP	CNP	F/CC	NOT RECOMMENDED
1 1 0	CNP	NC	F/CC	NOT RECOMMENDED
1 1 1	CNP	CNP	F/CC	NOT RECOMMENDED

SORC 27 26	ALU SOURCE OPERANDS R	 S
0 0	0	B*
0 1	I	B
1 0	I	Q
1 1	I	0

FUNC 22 21 20	FUNCTION	OPERATION DESCRIPTION
0 0 0	R+S	ADD
0 0 1	S-R	SUBTRACT (2's comp)
0 1 0	R-S	SUBTRACT (2's comp)
0 1 1	RvS	Logical inclusive OR
1 0 0	R∧S	Logical AND
1 0 1	$\bar{R}$∧S	Logical Complement AND
1 1 0	R⊕S	Logical Exclusive OR
1 1 1	$\overline{R\oplus S}$	Logical Exclusive NOR

H 28	OPERATION DESCRIPTION
0	NO SHIFT ON IMMEDIATE OPERAND
1	SHIFT IMMEDIATE OPERAND 16 BITS END-AROUND

Notes:

1) NC indicates NO CHANGE.

2) F indicates DATA OUTPUT OF ALU.

3) CC indicates ALU CONDITION CODES-N,V,C,Z.

4) CNP indicates UNPREDICTABLE.

*OPCODE 0 PROVIDES FASTER EXECUTION FOR SORC=0

Instruction Timing
ALL OPERATIONS 350ns

OPCODE 3 Read Program Memory (PM)

Description:

Provides for loading a 32-bit word of data from the program memory into any of the 16 GP registers. The register loaded is selected by the B field. If TAG=0, the effective address is taken from the DIRECT ADDRESS field. If TAG≠0, the effective address is the content (least significant 12 bits) of the GP register specified by the TAG field. The status register is loaded with the contents of the GP register specified by the TAG field (for all values of TAG=0 thru F).

Bus parity is checked at the end of a read instruction cycle; if an error is found, the CPU issues interrupt 5.

OPCODE 3 READ PROGRAM MEMORY (PM)

31 30 29	28 27 26	25 24 23	22 21 20	19 18 17 16	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
0 1 1	1 1 1	0 1 1	0 1 1	B	TAG	DIRECT ADDRESS

NOTES ON READ FORMAT: SORC=7, FUNC=3 must be chosen to provide the fastest means to load the GP registers from the bus and is the only means supported by the clock. If TAG ≠ O, DEST=3 must be chosen because this provides the fastest means to get GP register contents (effective address) out to the CPU bus and is the only means supported by the clock. If TAG=O, DEST=0-3 may be chosen; shifting is not supported by the clock. Register Q may be loaded (DEST=1) from the program memory only if TAG=O (direct addressing). With DEST=3, the status register is loaded with the contents of the GP register specified by the TAG field. If DEST≠3, the status register is loaded with the ALU output (PM data v O). Regardless of DEST, the four condition code bits of the status register will reflect the ALU output (PM data v O).

INSTRUCTION TIMING

ALL OPERATIONS 525ns

OPCODE 3 Write Program Memory (PM)

Description:

Provides for storing the 32-bit content of any of the 16 GP registers into the program memory. If TAG=0, the effective address is taken from the DIRECT ADDRESS field. If TAG≠0, the effective address is the content (least significant 12 bits) of the GP register specified by the TAG field. The source register is specified by the B field to be one of the 16 GP registers. The contents of the 16 GP registers and status register are not changed.

OPCODE 3 WRITE PROGRAM MEMORY (PM)

31 30 29	28 27 26	25 24 23	22 21 20	19 18 17 16	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
0 1 1	0 0 0	0 1 1	X X X	B	TAG	DIRECT ADDRESS

NOTES ON WRITE FORMAT: SORC must be chosen any value 0-4 to indicate to the clock that the instruction cycle is an outbound (write) operation. Source operands to the ALU are irrelevant because the ALU is being bypassed (DEST=3) and neither B,Q, nor status registers are being loaded (clock is inhibited to 2901 and status register). For the same reasons, the FUNC field is irrelevant. The DEST field must be chosen to be 3 because this provides the fastest means to get GP register contents out to the CPU bus and is the only means supported by the clock. The contents of the Q register can not be written to the program memory.

INSTRUCTION TIMING	
ALL OPERATIONS	650ns

OPCODE 4 External Function Control (EFC)

Description:

Provides for the control and/or interrogation of sixteen (16) bistable elements via the external function (EXF) logic. An external function code (8 bits wide consisting of the XFDA, NS and SR fields) is issued to the EXF logic during the instruction cycle. The external function device address (XFDA) field specifies the element to be controlled and/or interrogated. The new state (NS) field defines whether to set, clear, nop, or toggle the specified element.

The interrogation is achieved by returning a sense bit from the specified element to the program control unit (PCU). If the sense bit is one, the next instruction in sequence is skipped; if the sense bit is zero, the next instruction in sequence is executed. The generation of the sense bit is defined by the sense request (SR) field. If SR=1, the sense bit reflects the true state of the element (one if one, zero if zero); if SR=2, the sense bit reflects the complement state of the element (one if zero, zero if one). SR=0 or 3 generate the sense bit independent of the state of the element.

The external function code can both control and interrogate an element in the same instruction cycle. The sense bit generated is determined by the state of the element before it has made its transition to the new state.

OPCODE 4 EXTERNAL FUNCTION CONTROL (EFC)

31 30 29	28	27 26 25 24 23 22 21 20	19 18 17 16 15 14 13 12 11 10 9 8	7 6 5 4	3 2	1 0
1 0 0	X	RC	X X X X X X X X X X X X	XFDA	NS	SR

XFDA 7 6 5 4	SPECIFIED ELEMENT
0 0 0 0	INTERLOCK 0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	
1 0 0 1	
1 0 1 0	
1 0 1 1	
1 1 0 0	REPEAT COUNTER
1 1 0 1	MULTIPLY MODE FF
1 1 1 0	CS PTR
1 1 1 1	ACTIVE FF

NS 3 2	NEW STATE
0 0	CLEAR
0 1	NOP
1 0	TOGGLE
1 1	SET

SR 1 0	GENERATE SENSE
0 0	ZERO
0 1	ONE IF ONE
1 0	ONE IF ZERO
1 1	ONE

NOTES:

1) The EFC instruction must reside at an odd address in the program memory (program counter and sense bit OR'ed to achieve skip).

2) Two of the elements addressed by the XFDA field are not bistable elements - the repeat counter & condition stack pointer (CS PTR). The repeat counter provides for the repetitive execution of an instruction; the CS PTR is used in conjunction with a 4x36 RAM file to implement the status (register) stack. The repeat counter is an 8-bit counter whose value is normally zero. When an EFC instruction is executed to set this element, the value in the repeat count (RC) field is loaded into the counter, causing the PCU to repeat (execution of) the instruction following the EFC instruction for RC+2 iterations. The PCU repeat mode exits with the repeat counter back at zero. The CS PTR is a 2-bit counter which points to the word currently being accessed in the 4 word status stack. The CS PTR can be initialized to zero by an EFC clear instruction. The EFC set instruction will cause the CS PTR to decrement by one (typically, bumping the CS PTR would be done by OPCODE 5). The state of these counter elements will return a zero/nonzero value as the sense bit.

INSTRUCTION TIMING	
SKIP OR NO SKIP	350ns

OPCODE 5 Interrupt Control

Description:

Provides control of the Priority Interrupt Controller (PIC) and status (register) stack. The Interrupt Control Command (ICC) field defines the PIC operation. Many PIC operations involve the CPU registers. Loading and reading the mask register and status register within PIC is achieved by data transfers over the bus between the PIC and CPU. The source to the bus is specified by the ICC field. No direct path between PIC and PM is provided.

CPU operation is controlled by the A,B,SORC,DEST, and FUNC fields. Any two of the 16 GP registers may be specified via fields A and B. Two source operands to the ALU, the ALU function, and destination of the ALU output are specified by the SORC, FUNC, DEST fields respectively.

Bumping the status stack is controlled by the stack status command (SSC) field. If the SSC field specifies to push or pop the status stack and the DEST field specifies to load the status register (both in the same instruction cycle), the bump will precede the load. The status stack is 4 words deep; if the bump causes the stack to overflow or underflow (pop when empty), the CPU will issue interrupt #4.

OPCODE 5 INTERRUPT CONTROL

31 30 29	28 27 26	25 24 23	22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6	5 4 3 2 1 0
1 0 1	SORC	DEST	FUNC	B	A	ICC	SCC	X X X X X X

DEST 25 24 23	Load B Register	Load Q Register	Load Status Register	Output to BUS (if enabled by ICC)	OPERATION DESCRIPTION
0 0 0	NC	NC	NC	F	NOP
0 0 1	NC	F	F/CC	F	LOAD Q
0 1 0	F	NC	F/CC	F	LOAD B
0 1 1	F	NC	A/CC	A	LOAD B; LOAD STATUS WITH A
1 0 0	▲F/2	NC	F/CC	F	SHIFT B TOWARDS LSB
1 0 1	▲F/2	▲Q/2	F/CC	F	SHIFT B & Q TOWARDS LSB
1 1 0	▲ 2F	NC	F/CC	F	SHIFT B TOWARDS MSB
1 1 1	▲ 2F	▲Q/2	F/CC	F	SHIFT B & Q TOWARDS MSB

SORC	ALU SOURCE OPERANDS	
28 27 26	R	S
0 0 0	A	Q
0 0 1	A	B
0 1 0	0	Q
0 1 1	0	B
1 0 0	0	A
1 0 1	*BUS	A
1 1 0	*BUS	Q
1 1 1	*BUS	0

FUNC 22 21 20	FUNCTION	OPERATION DESCRIPTION
0 0 0	R+S	ADD
0 0 1	S-R	SUBTRACT (2's comp)
0 1 0	R-S	SUBTRACT (2's comp)
0 1 1	$R \vee S$	Logical inclusive OR
1 0 0	$R \wedge S$	Logical AND
1 0 1	$\bar{R} \wedge S$	Logical complement AND
1 1 0	$R \oplus S$	Logical exclusive OR
1 1 1	$\overline{R \oplus S}$	Logical exclusive NOR

▲ Logical zero shift only; if SORC=5-7, result is unpredictable.

* Valid only if ICC=6 or 7

INTERRUPT CONTROL COMMAND 11 10 9 8	SOURCE TO BUS	OPERATION DESCRIPTION
0 0 0 0	NONE	MASTER CLEAR
0 0 0 1	NONE	CLEAR ALL INTERRUPTS
0 0 1 0	CPU	CLEAR INTERRUPTS FROM BUS
0 0 1 1	NONE	CLEAR INTERRUPTS FROM MASK REGISTER
0 1 0 0	NONE	CLEAR INTERRUPT, LAST VECTOR READ
0 1 0 1	NONE	READ VECTOR; LOAD STATUS REGISTER TO V+1
0 1 1 0	PIC	READ STATUS REGISTER
0 1 1 1	PIC	READ MASK REGISTER
1 0 0 0	NONE	SET MASK REGISTER
1 0 0 1	CPU	LOAD STATUS REGISTER
1 0 1 0	CPU	BIT CLEAR MASK REGISTER
1 0 1 1	CPU	BIT SET MASK REGISTER
1 1 0 0	NONE	CLEAR MASK REGISTER
1 1 0 1	NONE	DISABLE INTERRUPT REQUESTS
1 1 1 0	CPU	LOAD MASK REGISTER
1 1 1 1	NONE	ENABLE INTERRUPT REQUESTS

SSC 7 6	STATUS STACK OPERATION
0 0	NC
0 1	NC
1 0	PUSH
1 1	POP

NOTES:

1) NC indicates NO CHANGE
2) F indicates DATA OUTPUT OF ALU.
3) CC indicates ALU CONDITION CODES-N,V,C,Z.
4) X indicates DON'T CARE

INSTRUCTION TIMING
ALL OPERATIONS 400ns

OPCODE 6 PC Stack Control

Description:

Transfers program control to the instruction pointed to by the jump select (JS) field. The JS field defines the effective address (EA); for all values of JS, the next instruction is fetched at EA and the PC is loaded with EA+1.

JS=2 causes the current contents of the PC (will contain address of next instruction in sequence) to be pushed onto the PC stack; EA is defined to be the DIRECT ADDRESS field of the instruction.

JS=3 defines the EA to be the PC stack; at the end of the instruction cycle, the PC stack is popped.

JS=1 defines the EA to be the contents of the Interrupt Vector Hold Register (IVHR); the PC stack is unchanged.

JS=0 is used to pop the PC stack; the EA is defined to be the current contents of the PC.

OPCODE 6 PC STACK CONTROL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11 10 9 8 7 6 5 4 3 2 1 0
1	1	0	X	X	X	JS		X	X	X	X	X	X	X	X	X	X	X	X	DIRECT ADDRESS

JS 25 24	SOURCE OF EFFECTIVE ADDRESS	BUMP PC STACK	OPERATION DESCRIPTION
0 0	PC	POP	POP PC STACK
0 1	IVHR	NC	JUMP TO INTERRUPT HANDLER
1 0	DIRECT ADDRESS	PUSH	JUMP TO SUBROUTINE
1 1	PC STACK	POP	RETURN FROM SUBROUTINE

NOTES:

1) NC indicates NO CHANGE
2) X indicates DON'T CARE
3) PC indicates PROGRAM COUNTER

INSTRUCTION TIMING

ALL OPERATIONS 300ns

OPCODE 7 Conditional Branch

Description:

Transfers program control conditional upon status register bits (V,N,Z,C,0-31). Eighty different tests are encoded by the CC SEL, REF, TS and BIT SEL fields. If the test condition is satisfied, instruction execution branches to the instruction pointed to by the DIRECT ADDRESS field. If the test condition is unsatisfied, instruction execution continues with the next instruction in normal sequence.

Either condition code tests or bit tests are enabled by the test select (TS) bit. Condition code select (CC SEL) chooses one of eight tests on condition code; branch occurs if test output is equal reference (REF) bit. Bit select (BIT SEL) chooses one of thirty-two (0-31) bits in the status register; branch occurs if selected bit value equals reference bit.

OPCODE 7 CONDITIONAL BRANCH

31 30 29	28	27 26 25	24	23 22	21	20 19 18 17 16	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
1 1 1	X	CC SEL	REF	X X	TS	BIT SEL	X X X X	DIRECT ADDRESS

20	19	18	17	16	SELECTED BIT
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	2
0	0	0	1	1	3
0	0	1	0	0	4
		•			•
		•			•
		•			•
1	1	1	1	0	30
1	1	1	1	1	31

21	TEST SELECT
0	CC SEL/REF
1	BIT SEL/REF

CC SEL				TEST OUTPUT (BRANCH IF EQUAL REF)	LOGICAL CONDITION FOR BRANCH (BRANCH ON TRUE)	OPERATION DESCRIPTION	
27	26	25	24				
0	0	0	0	0	1	UNCONDITIONAL BRANCH	
0	0	0	1	0	0	NOP (NO BRANCH)	
0	0	1	0	V	$\overline{V}$	BRANCH ON NO OVERFLOW	
0	0	1	1	V	V	BRANCH ON OVERFLOW	
0	1	0	0	N	$\overline{N}$	BRANCH ON POSITIVE	
0	1	0	1	N	N	BRANCH ON NEGATIVE	
0	1	1	0	Z	$\overline{Z}$	BRANCH ON NOT EQUAL (ZERO)	
0	1	1	1	Z	Z	BRANCH ON EQUAL (ZERO)	
1	0	0	0	C	$\overline{C}$	BRANCH ON NO CARRY; HIGHER OR SAME	▲
1	0	0	1	C	C	BRANCH ON CARRY; LOWER	▲
1	0	1	0	$V \oplus N$	$V \oplus \overline{N}$	BRANCH ON GREATER OR EQUAL (ZERO)	*
1	0	1	1	$V \oplus N$	$V \oplus N$	BRANCH ON LESS THAN (ZERO)	*
1	1	0	0	$\overline{(V \oplus N) \vee Z}$	$(V \oplus N) \vee Z$	BRANCH UN LESS OR EQUAL (ZERO)	*
1	1	0	1	$\overline{(V \oplus N) \vee Z}$	$(\overline{V \oplus N}) \wedge \overline{Z}$	BRANCH ON GREATER THAN (ZERO)	*
1	1	1	0	$\overline{Z \vee C}$	$Z \vee C$	BRANCH ON LOWER OR SAME	▲
1	1	1	1	$\overline{Z \vee C}$	$\overline{Z} \wedge \overline{C}$	BRANCH ON HIGHER	▲

∧ - AND
∨ - INCLUSIVE OR
⊕ - EXCLUSIVE OR
* - SIGNED 2's COMPLEMENT ARITHMETIC ASSUMED
▲ - UNSIGNED ARITHMETIC ASSUMED

Instruction Timing	
BRANCH CONDITION NOT MET	200ns
BRANCH CONDITION MET	300ns

SECTION IV

THE DISPLAY PROCESSOR

1. FUNCTIONAL DESCRIPTION

The Display Processor (DP) performs two separate functions in the feasibility model. These are referred to as the application and support roles. During operation as a passive detection system the application responsibility is to display a list of emitter beams currently active in the environment. This list can be presented via the system CRT. The source for this list is the communication processor which reports to the DP both new emitters found by the slaves and old emitters deleted from the preprocessor file. For the feasibility model development, the DP consists of a PDP-11V03 minicomputer supported by a CRI line printer and a dual floppy disc drive. The hardware interface between the DP and the multiprocessor is shown in Figure 18.

In a full scale operational system the DP responsibility would be expanded to include such functions as scan rate determination, ranging, identifying, threat evaluation and passive/active system control. Such expansion would most easily be accomodated by partitioning the tasks among several processors which are incorporated into the multiprocessor structure. Although no single task above represents a severe processing load from the standpoint of input rates or response time it is anticipated that the total load is sufficient to warrant the use of more than one processor.

The support function of the DP is as a program development aid and a diagnostic tool. In addition to the line printer and CRT mentioned above, a dual floppy disc and card reader interface are used to support this mode of operation.

The program development tool consists of a set of routines which run on the DP and effect interrogation (and modification) of the global memory (communication bank), any program memory, and the various registers within any of the microprocessors. These routines are collected into a program called MAPAID which is further explained below.

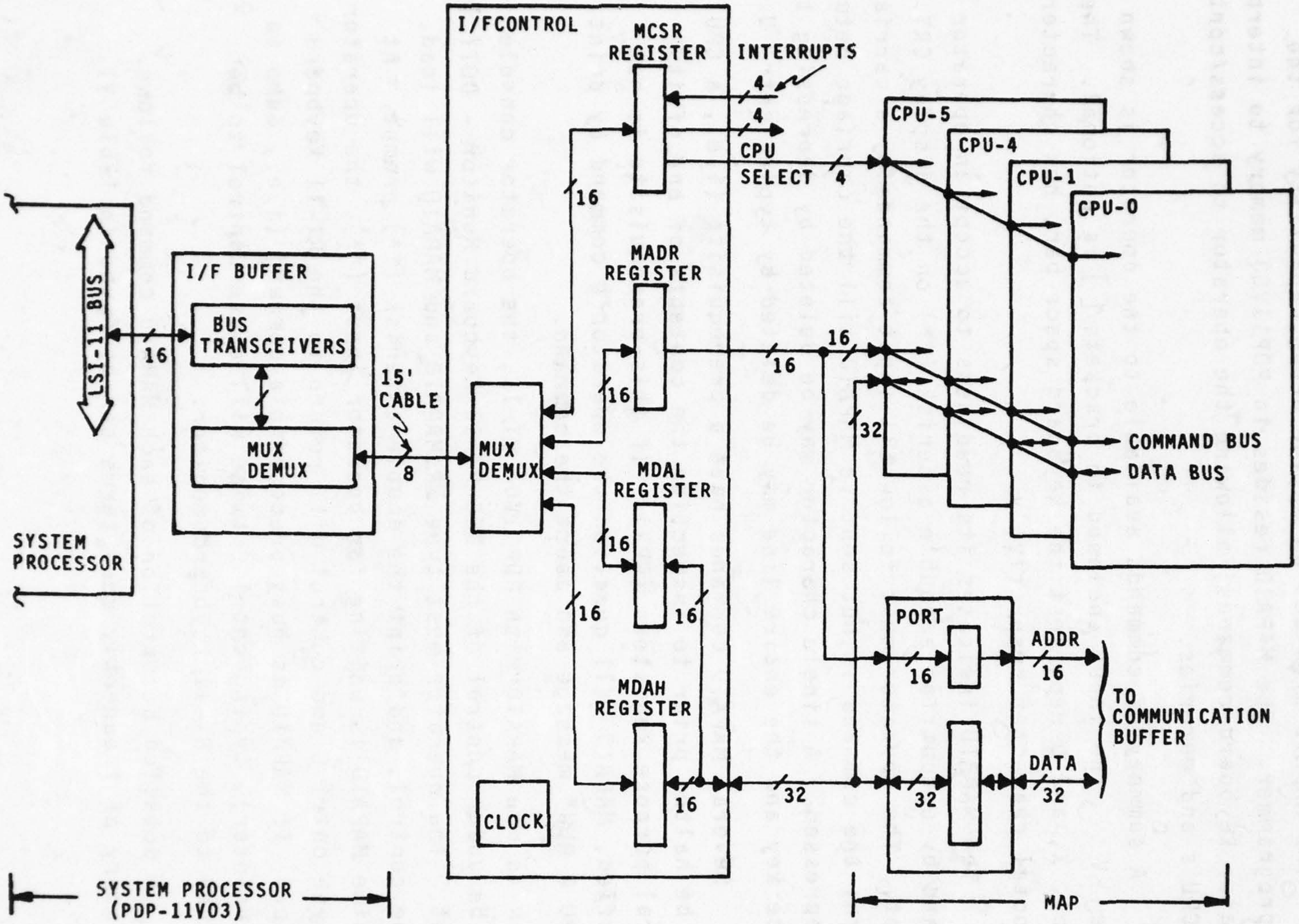


Figure 18 - Multiprocessor/Display Processor Interface Functional Block Diagram

The MAPAID provides an online debug capability for the MAP programmer. The MAPAID resides in PDP11V03 memory to interpret operator keyboard commands, allowing the operator to access/control MAP CPU's and memories.

A summary of commands available to the operator is shown in Table V. The input enclosed in brackets [ ] is optional. The Δ symbol, X, and O represent the keyboard space bar, hex character, and octal character respectively.

The MAPAID indicates its readiness to accept an operator command by outputting a double asterisk (**) on the display CRT prompt. The operator must follow all input commands by a carriage return; the command is not sent to MAPAID till the carriage return is depressed. A single character may be deleted by depressing the delete key and the entire line may be deleted by typing cntrl U.

Several MAPAID commands have a prerequisite (i.e., a CPU must be halted prior to inspecting the contents of one of its general purpose registers (GPR). If this prerequisite is not satisfied, MAPAID will question the operator's command by printing a "CPU @ RUN" message and abort the command.

To use MAPAID with PDP11V03 RT-11, the operator console must be under control of the DEC RT-11 Keyboard Monitor - DOT(.) prompt. The operator must type .RΔMAPAID and MAPAID will load, assume control, and print the double asterisk (**) prompt. At any time MAPAID is waiting for operator input (**), the operator may type cntrl C and control will return to the RT-11 keyboard monitor. If MAPAID is busy processing a command (i.e., dump to line printer), typing cntrl C twice will cause control to be returned to the RT-11 keyboard monitor.

A detailed description of each MAPAID command follows. A glossary of frequently used terms can be found in Table VI.

Table V

MAPAID Command Summary

COMMAND FORMAT	NOTES	FUNCTION	PAGE
CP[U] [ΔX]	-	SELECT/CONNECT CPU	5
HA[LT]	1)	HALT EXECUTION	6
ST[ART]ΔXXX	2)	START EXECUTION AT PM ADDRESS XXX	7
P[ROCEED]	2)	CONTINUE EXECUTION	8
RE SET	1)	INITIALIZE CPU	9
SE	2)	ENABLE SINGLE INSTRUCTION MODE	10
SD	2)	DISABLE SINGLE INSTRUCTION MODE	11
WA[IT] [ΔXXX]	1)	WAIT FOR CPU TO HALT	12
[DEV:]/R[ΔXX -XX]]	2)	DUMP,INSPECT/CHANGE CPU GP REGISTERS	13
[DEV:]/Q	2)	INSPECT/CHANGE CPU Q REGISTER	13
[DEV:]/I	2)	DUMP IR, PC, & PS	15
LPΔ[DEV:]FILNAM[.EXT]	2)	LOAD PM FROM DISKETTE	16
[DEV:]/P[ΔXXX[-XXX]]	2)	DUMP, INSPECT/CHANGE PM	17
LGΔ[DEV:]FILNAM[.EXT]	-	LOAD GM FROM DISKETTE	18
[DEV:]/G[ΔXXXX -XXXX]]	-	DUMP, INSPECT/CHANGE GM	19
[DEV:]/DΔ00000-00000	-	DUMP LSI-11 MEMORY	20

1) CPU MUST PREVIOUSLY BE CONNECTED.
2) CPU MUST PREVIOUSLY BE CONNECTED AND HALTED.

[] -	OPTIONAL INPUT
Δ -	KEYBOARD SPACE BAR
X -	HEX CHARACTER
0 -	OCTAL CHARACTER

Table VI

MAPAID Symbol Definitions

[]	OPTIONAL INPUT
Δ	KEYBOARD SPACE BAR
X	HEX CHARACTER
O	OCTAL CHARACTER
PM	PROGRAM MEMORY
GM	GLOBAL MEMORY
GPR	GENERAL PURPOSE REGISTERS (RO-RF, & RQ of CPU)
IR	INSTRUCTION REGISTER
PC	PROGRAM COUNTER
PS	PROGRAM (COUNTER) STACK

1) CP[U][△X] SELECT/CONNECT CPU

ARG X specifies CPU to be connected to operator console. After making connection, MAPAID responds by printing CPU run/halt status and PM size. If ARG X is omitted, command is assumed to be an operator inquiry asking which CPU is currently connected.

2) HA[LT] HALT EXECUTION

This command is used to stop execution of the selected CPU. The MAPAID responds by returning a PM address (NNNN) to the operator's console. The instruction at this PM address has not been executed yet and will be the first instruction to be executed if execution is continued. The instruction at NNNN has already been loaded in the IR and the PC contains NNNN+1. If the halt address message at the operator's console is preceded by a question mark, the CPU was already in the halt state when the input command was received.

3) ST[ART]△XXX START EXECUTION AT PM ADDRESS XXX

This command specifies to a CPU to begin execution at PM address XXX. A CPU must have been previously selected and halted.

4) P[ROCEED] CONTINUE EXECUTION

This command specifies to a CPU to continue execution using the current PC and IR contents. A CPU must have been previously selected and halted.

5) RE[SET] INITIALIZE CPU

This command will cause initialization of the selected CPU. Initialization consists of executing instructions at PM addresses 5, 6, & 7 to achieve a master clear to the 2914 interrupt logic and asserting the hardware master reset line.

5) RE[SET] INITIALIZE CPU (continued)

Upon completion of the reset, the CPU will be halted at PM address 7. The halt/enable flipflop will be at halt The interlocks, repeat counter, condition stack pointer, and multiply mode flipflop will all be cleared to zero. All sight interrupts will be cleared; the interrupt mask register, interrupt status register, and interrupt enable flipflop are all cleared making the CPU ready to respond to any interrupt.

Single instruction mode will be maintained if enabled. An alternative command format is X followed by carriage return.

6) SE ENABLE SINGLE INSTRUCTION MODE

This command puts the selected CPU into the single instruction mode of operation. In this mode, each time program execution is initiated (via start or proceed command), the selected CPU executes one instruction cycle and MAPAID responds by returning the halt address to the operator console.

7) SD DISABLE SINGLE INSTRUCTION MODE

This command terminates the single instruction mode of operation.

8) WA[IT][ΔXXX] WAIT FOR CPU TO HALT

This command is intended to facilitate operation of MAPAID under batch control. The WAIT command does not return control (the prompt) to the operator console until the selected CPU has halted. If the ARG XXX is specified and the batch handler (BA) is loaded (in PDP11V03 memory), the CPU PM halt address and ARG XXX are compared. If they are equal, batch variable "G" is cleared, set to -1 otherwise. If ARG XXX is omitted from command string, batch variable "G" is unchanged.

9) [DEV:]/R[ΔXX_1[-XX_2]] DUMP,INSPECT/CHANGE CPU GP REGISTERS.

If both arguments, XX_1 and XX_2, are present, the command signifies a dump of the CPU GP registers. XX_1 specifies the first GPR to dump; XX_2 specifies the last GPR to dump. The dump is made to the operator console by default but the command may be preceded by a device dataset steering the dump to the line printer.

If only the first argument, XX_1, is present, the command signifies an inspect/change of one CPU GPR. The argument specifies the register to be opened (display contents). A carriage return (CR) closes the register; a line feed (LF) closes the register and opens the next one; an up arrow (↑) closes the register and opens the previous one. The CR, LF, or UP ARROW, either one, may be preceded by a 32-bit hex argument, which would be deposited into the open register before it's closed.

If no arguments follow the /R command, the last register closed is reopened.

An inspect $ change on CPU register RQ may be requested by the /Q command (no argument) or by inputting Hex 10 for the /R argument XX_1.

When the /R (or /Q) command is used to change the contents of a CPU GPR, the condition codes are loaded accordingly to reflect the new register contents. Inspection of the CPU GPR contents does not affect the condition codes.

The CPU must have been previously selected and halted.

10) [DEV:]/I DUMP IR, PC & PS

This command causes the current contents of the instruction register (IR), program counter (PC), and top address of the program stack (PS) to be printed to the operator's console. The command may be preceeded by a device dataset steering the dump to the line printer. The CPU must have been previously selected and halted.

11) LP△[DEV:]FILNAM[.EXT]   LOAD PM FROM DISKETTE

This command signifies to load the program memory from the diskette file specified in the argument. The argument is a standard DEC device dataset. Default device is DK, default extension is MEC. The CPU must have been previously selected and halted.

Diskette file format shown in Figure 19.

12) [DEV:]/P[△XXX$_1$ -XXX$_2$]]   DUMP, INSPECT/CHANGE PM

If both arguments, XXX$_1$ and XXX$_2$, are present, the command signifies a dump of program memory (PM). The XXX$_1$ specifies the first PM address to dump; XXX$_2$ specifies the last PM address to dump. The dump is made to the operator console by default but the command may be preceeded by a device dataset steering the dump to the line printer or diskette (file format at Figure 19).

If only the first argument, XXX$_1$, is present, the command signifies an inspect & change of one PM address. The argument specified the PM location to be opened (display contents). A carriage return (CR) closes the location; a line feed (LF) closes the location and opens the next one; an up arrow (↑) closes the location and opens the previous one. The CR, LF, or up arrow, either one, may be preceded by a 32-bit hex argument, which would be deposited into the open location before it's closed.

If no arguments follow the /P command, the last PM location closed is reopened.

The CPU must have been previously selected and halted.

13) LG△[DEV:]FILNAM[.EXT]   LOAD GM FROM DISKETTE

This command signifies to load the global memory from the diskette file specified in the argument. The argument is a standard DEC device dataset. Default device is DK, default extension is MEC. Diskette file format shown in Figure 19.

14) [DEV:]/G[ΔXXXX$_1$[-XXXX$_2$]]   DUMP, INSPECT/CHANGE GM

If both arguments, XXXX$_1$ and XXXX$_2$, are present, the command signifies a dump of global memory (GM). XXXX$_1$ specifies the first GM address to dump; XXXX$_2$ specifies the last GM address to dump. The dump is made to the operator console by default, but the command may be preceded by a device dataset steering the dump to the line printer or diskette (file format at Figure 19).

If only the first argument, XXXX$_1$ is present, the command signifies an insert & change of one GM address. The argument specifies the GM location to be opened (display contents). A carriage return (CR) closes the location; a line feed (LF) closes the location and opens the next one; an up arrow (↑) closes the location and opens the previous one. The CR, LF, or up arrow, either one, may be preceded by a 32-bit hex argument, which would be deposited into the open location before it's closed.

If no arguments follow the /G command, the last GM location closed is reopened.

15) [DEV:]/DΔ00000$_1$-00000$_2$ DUMP LSI-11 MEMORY

This command signifies a dump of LSI-11 memory. Unlike other MAPAID commands, input arguments and output dumps are in octal format. The 00000$_1$ specifies the first LSI-11 address to dump; 00000$_2$ specifies the last LSI-11 address to dump. The dump is made to the operator console by default, but the command may be preceded by a device dataset steering the dump to the line printer.

Dump to diskette or inspect and change of LSI-11 memory are provided by DEC system software. Dump to diskette is provided by the SAVE command to the keyboard monitor; inspect and change is provided by the console ODT microcode or ODT software utility.

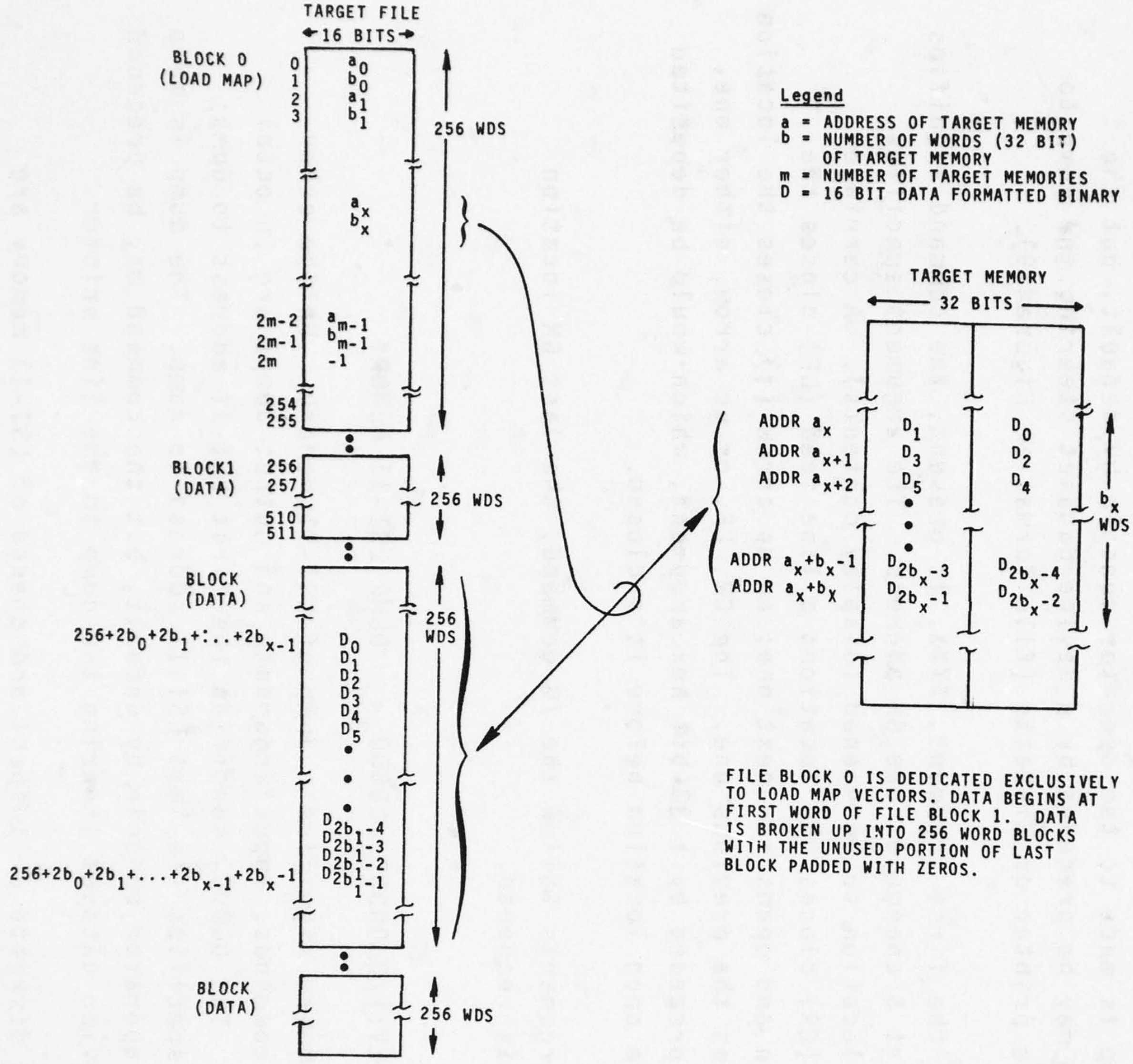


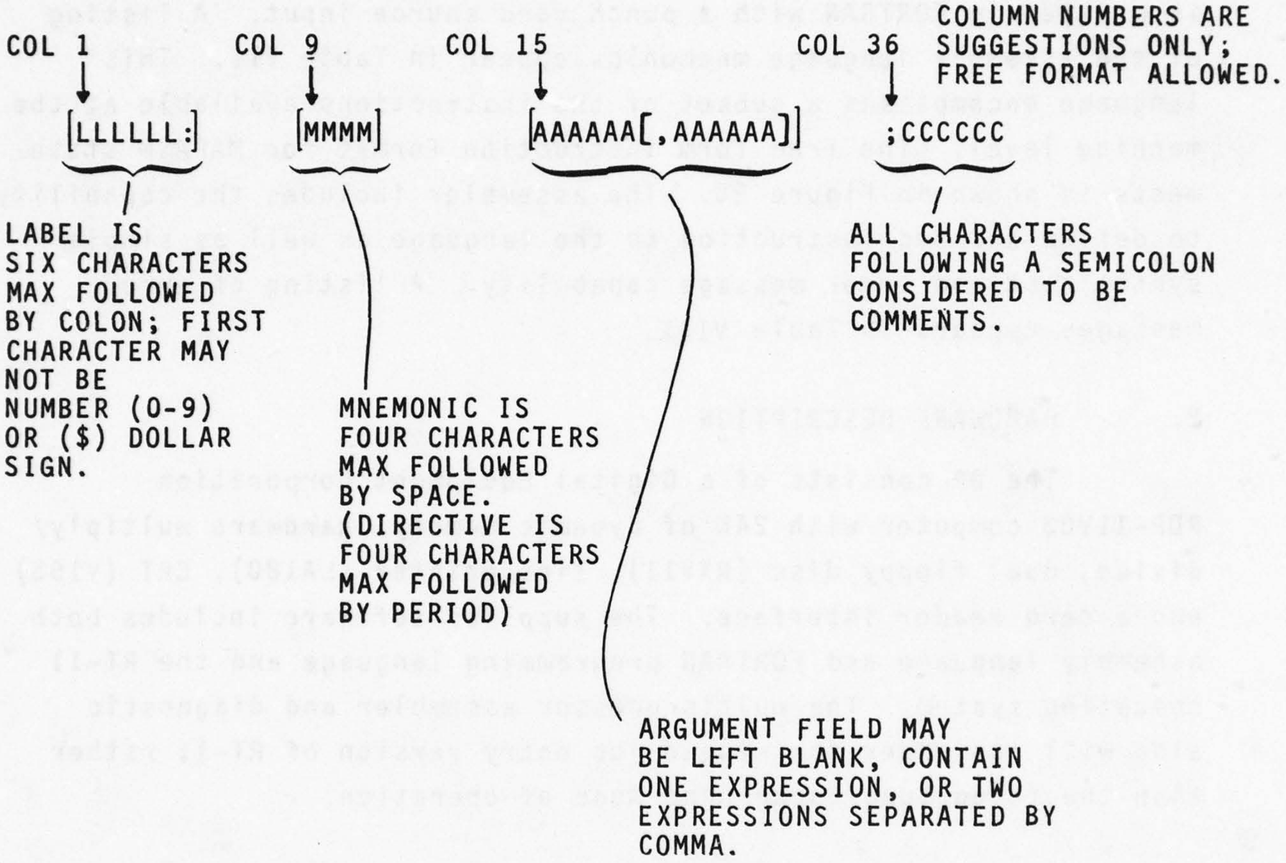
Figure 19 - Format of Diskette File Associated with MAPAID Commands LP,LG,/P, and/G.

The program development function of the DP is also supported with a simple assembler (termed MAPASM) for the multiprocessor. The purpose of the MAPASM is to expedite the writing of application software for the multiprocessor. The assembler is written in FORTRAN with a punch card source input. A listing of the assembly language mnemonics appear in Table VII. This language encompasses a subset of the instructions available at the machine level. The free form instruction format for MAPASM statements is shown in Figure 20. The assembler includes the capability to define and add instruction to the language as well as simple syntax test and error message capability. A listing of error messages appears in Table VIII.

2. HARDWARE DESCRIPTION

The DP consists of a Digital Equipment Corporation PDP-11V03 computer with 24K of dynamic memory, hardware multiply/divide, dual floppy disc (RXV11), line printer (LA180), CRT (V155) and a card reader interface. The supplied software includes both assembly language and FORTRAN programming language and the RT-11 operating system. The multiprocessor assembler and diagnostic aids will run under the single job entry version of RT-11 rather than the foreground/background mode of operation.

[ ] BRACKETS INDICATE OPTIONAL



COL 1 COL 9 COL 15 COL 36

COLUMN NUMBERS ARE SUGGESTIONS ONLY; FREE FORMAT ALLOWED.

[LLLLLL:] [MMMM] AAAAAA[,AAAAAA]] ;CCCCCC

LABEL IS SIX CHARACTERS MAX FOLLOWED BY COLON; FIRST CHARACTER MAY NOT BE NUMBER (0-9) OR ($) DOLLAR SIGN.

MNEMONIC IS FOUR CHARACTERS MAX FOLLOWED BY SPACE. (DIRECTIVE IS FOUR CHARACTERS MAX FOLLOWED BY PERIOD)

ARGUMENT FIELD MAY BE LEFT BLANK, CONTAIN ONE EXPRESSION, OR TWO EXPRESSIONS SEPARATED BY COMMA.

ALL CHARACTERS FOLLOWING A SEMICOLON CONSIDERED TO BE COMMENTS.

AN ARITHMETIC EXPRESSION (EXP) MAY BE ANY COMBINATION OF THE FOLLOWING WHOSE FINAL VALUE LIES WITHIN A 16 BIT RANGE (-32767 TO +32767):

+,-,* ARITHMETIC OPERATORS (* EVALUATED FIRST, THEN LEFT TO RIGHT)
LABELS
DECIMAL NUMBERS (-32767 TO +32767)
$ INDICATING PRESENT ADDRESS
RO THRU R15 INDICATING CPU GP REGISTERS

ON LISTING OUTPUT, ERROR MESSAGES PRECEED TARGET SOURCE LINE.

Figure 20 - MAPASM Instruction Format

Table VII

Multiprocessor Assembly Language Mnemonics

MNEMONIC	COMMENT
CLR	CLEAR REGISTER
LRR	LOAD REGISTER TO REGISTER
COM	COMPLEMENT REGISTER
INC	INCREMENT REGISTER
DEC	DECREMENT REGISTER
NEG	NEGATE REGISTER
ADD	ADD REGISTER TO REGISTER
SUB	SUBTRACT REGISTER TO REGISTER
AND	LOGICAL AND
XOR	EXCLUSIVE OR
ASR	ARITHMETIC SHIFT REGISTER RIGHT
ASL	ARITHMETIC SHIFT REGISTER LEFT
ROR	ROTATE REGISTER RIGHT
ROL	ROTATE REGISTER LEFT
MPQ	MULTIPLY BY Q REGISTER VALUE
LQR	LOAD Q FROM REGISTER
LRQ	LOAD REGISTER FROM Q
NOP	NO OPERATION
RD	READ EXTERNAL DEVICE
WD	WRITE EXTERNAL DEVICE
WE,+	WRITE EXTERNAL DEVICE AND INCREMENT REGISTER
WD,-	WRITE EXTERNAL DEVICE AND DECREMENT REGISTER
LI	LOAD IMMEDIATE
LIH	LOAD IMMEDIATE HIGH HALF WORD
AI	ADD IMMEDIATE
SI	SUBTRACT IMMEDIATE
CI	COMPARE IMMEDIATE
LD	LOAD REGISTER FROM PROGRAM MEMORY (PROGRAM MEMORY ADDRESS IN INSTRUCTION)

Table VII (cont.)

MNEMONIC	COMMENT
LDX	LOAD REGISTER FROM PROGRAM MEMORY INDEXED (PROGRAM MEMORY ADDRESS IN A REGISTER)
LDQ	LOAD Q FROM PROGRAM MEMORY
ST	STORE REGISTER IN PROGRAM MEMORY
STX	STORE REGISTER IN PROGRAM MEMORY INDEXED
RPT	REPEAT NEXT INSTRUCTION N TIMES
HALT	SUSPEND EXECUTION
EXF	MANIPULATE EXTERNAL FUNCTION FLAG
JSR	JUMP TO SUBROUTINE
RTS	RETURN FROM SUBROUTINE
BBC	BRANCH IF BIT CLEAR
BBS	BRANCH IF BIT SET
BR	BRANCH UNCONDITIONALLY
BVC	BRANCH IF OVERFLOW IS CLEAR
BVS	BRANCH IF OVERFLOW IS SET
BPL	BRANCH IF PLUS
BMI	BRANCH IF MINUS
BNE	BRANCH IF NOT EQUAL (TO ZERO)
BEQ	BRANCH IF EQUAL (TO ZERO)
BCC	BRANCH IF CARRY IS CLEAR
BCS	BRANCH IF CARRY IS SET
BGE	BRANCH IF GREATER THAN OR EQUAL
BLT	BRANCH IF LESS THAN (ZERO)
BLE	BRANCH IF LESS THAN OR EQUAL (TO ZERO)
BGT	BRANCH IF GREATER THAN (ZERO)
BLOS	BRANCH IF LOWER OR SAME
BHI	BRANCH IF HIGHER
BHIS	BRANCH IF HIGHER OR SAME
BLO	BRANCH IF LOWER

Table VII (end)

MNEMONIC	COMMENT
	DIRECTIVE
DC.	DEFINE CONSTANT
DS.	DEFINE STORAGE
EQU.	EQUATE
PAGE	MOVE TO TOP OF PAGE
EVEN.	LOCATE INST AT AN EVEN ADDRESS
ODD.	LOCATE INST AT AN ODD ADDRESS
ORG.	DEFINE FIRST PROGRAM MEMORY ADDRESS
END.	END OF SOURCE
NAME.XYX	PROGRAM NAME
GEN.	GENERATE NEW OPCODE (DEFINE NEW INSTRUCTION)

Table VIII

MAPASM Error Key

A - ARGUMENT MISSING
C - CONSTANT COULDN'T BE EVALUATED
D - DOUBLE DEFINED LABEL
E - SOURCE LINE FORMAT ERROR
F - FIRST CHARACTER IN LABEL ILLEGAL
G - LENGTH OF EXPRESSION ELEMENT EXCESSIVE
L - LENGTH OF LABEL OR MNEMONIC EXCESSIVE
M - MNEMONIC UNDEFINED
O - ORG STATEMENT MISSING; PROGRAM CAN NOT BE LOADED
S - END STATEMENT MISSING; STATEMENT APPENDED
U - UNDEFINED SYMBOL IN ARGUMENT FIELD
V - EXPRESSION VALUE EXCEEDS FIELD WIDTH ALLOCATION
W - UNDEFINED DIRECTIVE

SECTION V

CONCLUSIONS AND RECOMMENDATIONS

1. CONCLUSIONS

The purpose of this effort was to establish the practicality of multiprocessor systems for Electronic Warfare application. The goal was to develop system concepts which would provide substantial increases in system throughput without sacrificing the flexibility of a conventional computer system.

Based on bench testing of the feasibility model, it can be concluded that it is practical to construct multiprocessor based systems which can analyze from several hundred thousand to over a million radar intercepts per second. The lower end of this range can be directly supported by the feasibility model and requires execution of approximately 20 million instructions per second among its various processors to derive and track pulse trains. The processing system occupies approximately one cubic foot. The upper end of the performance range requires increasing the number of microprocessors in the multiprocessor and repackaging the preprocessor to support a higher clock rate. Typical projected system input pulse rates and multiprocessor subsystem rates are shown in Table IX, for various numbers of microprocessors in the multiprocessor subsystem. These numbers are projections rather than actual measurements because of the bench test equipment provided very limited signal generation capability. However, short bursts of over 200,000 intercepts per second were obtained for testing. The projections of performance were based on observations of processors idle time for sparse environments and close agreement between the model performance and simulation prediction performed under ESM HYBRID PROCESSING TECHNIQUES DEVELOPMENT (Contract F33615-74-C-1101).

TABLE IX

PROJECTED PROCESSING RATES

NUMBER OF MICROPROCESSORS	MULTIPROCESSOR SUBSYSTEM INPUT PULSE RATE (INTERCEPTS/SECOND)	SYSTEM INPUT PULSE RATE (INTERCEPTS/SECOND)
2	24,000	240,000-480,000
3	42,000	420,000-840,000
4	57,000	570,000-1,140,000
5	69,000	690,000-1,380,000
6	77,000	770,000-1,540,000
7	82,000	820,000-1,640,000
8	86,000	860,000-1,760,000

This development program has also demonstrated that the two classical multiprocessor drawbacks do not form a hinderance for Electronic Warfare applications. These problems are treated below.

The first major hurdle to overcome in any multiprocessor implementation is devising a method of synchronization for communication among the various independent subsystems. In general, when arranging for intercommunication between subsystems that do not share a common time reference, it is impossible to avoid generation of signals that are not logically defined during sampling by one or the other subsystems. Discussions at the Workshop on Synchronizer Failure (Washington University, St. Louis, MO; April 27-28, 1972) has revealed that a number of computer systems of various manufacturers are subject to significant rates of system failures resulting from unrealiable interaction between mutually asychronous subsystems. The popular solution to this problem is to lower the sampling rate (thereby reducing the number of failures per unit time) and providing a means for system recovery. Whereas this approach is adequate for low speed system peripherals, it is not viable for multiprocessors configured to maximize throughput. Therefore, an economic means had to be devised to support extremely high interrogation rates between subsystems without synchronizing failures. For example, in the feasibility model there are 9 independently clocked subsystems which must interact with each other. These subsystems include; the 4 microprocessors, the pre-processor, the 3 global memory controllers and the display processor. Individual interrogation rates between various subsystem pairs range from thousands to millions of interrogations per second. At the system level this translates into 10's of millions of interrogations per second. Prolonged operation of the feasibility model has clearly demonstrated that the very high subsystem interaction rates can be supported in a cost effective manner through proper hardware design.

The second hurdle concerning multiprocessor use involves maintaining software coordination among the various processors. Simply stated "The cost of developing a complex operating system capable of synchronizing a number of processor operating simultaneously has been demonstrated over the years to be overwhelming". (Computer/Processors [for Electronic Warfare] NRL Report 8247, 15 Aug. 1978). Although the MAP effort did not attack the broad problem of multiprocessor operating systems for general applications; it did determine that for dedicated use in a master/ slave mode maintaining control and coordination among the various processors is not an overly difficult task. This fact is best exemplified by examination of the slave control executed by the master (see Figure 14) which require less than 1000 instructions. The slaves on the otherhand have less than 10 percent of their code devoted to the mechanics of interprocessor control and coordination. It is true however that communication protocal must be carefully structured to avoid loss of coordination or excessive overhead time penalities.

2. RECOMMENDATIONS

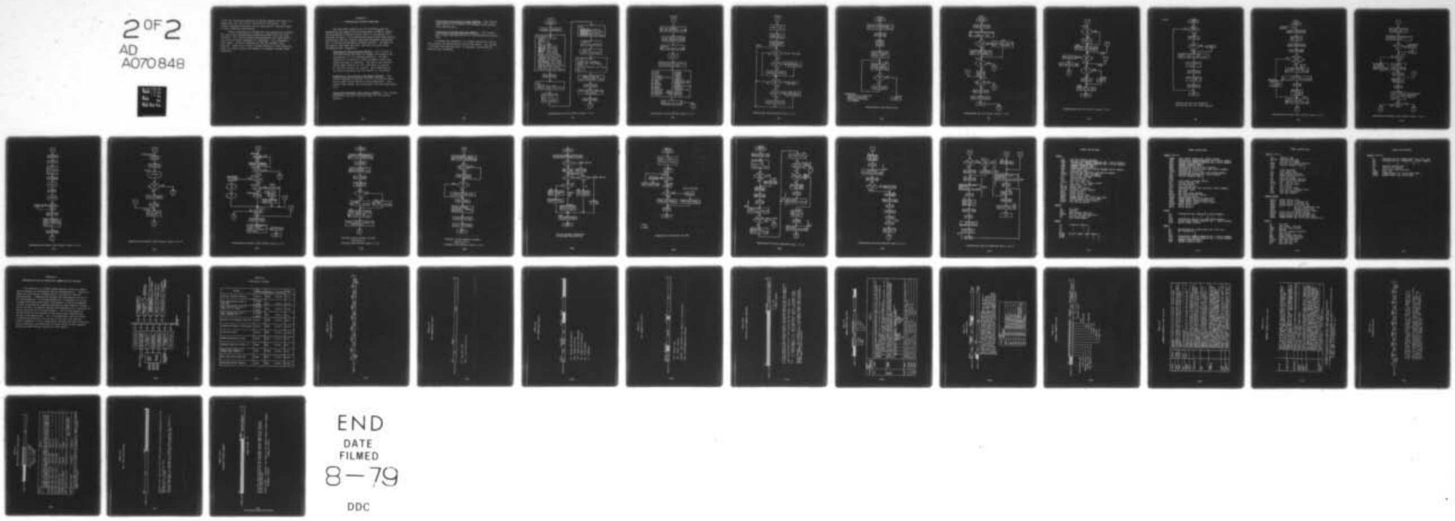
It is suggested that this beginning work be carried forward in the following three areas:

First, it is recommended that bench testing be continued with the use of denser simulated radar environments. This work would further validate performance projections. It would also allow the study of subsystem interaction when subjected to higher input data rates which more closely reflect real world conditions. Of particular interest would be the dynamic behavior of data buffers passed between subsystems of greatly different processing speeds.

Second, the facility for the multiprocessor architecture to handle exotic emitter types should be studied. This area of investigation looks promising because the multiprocessor is well

AD-A070 848 GOODYEAR AEROSPACE CORP AKRON OHIO F/G 9/2
MICROCOMPUTER ARRAY PROCESSOR.(U)
OCT 78 R H RIES, R A HUJAR, F C CARTY F33615-75-C-1179
UNCLASSIFIED GER-16565 AFAL-TR-78-157 NL

2 OF 2
AD
A070848



END
DATE
FILMED
8-79
DDC

suited for concurrent execution of several complex algorithms on a common data set with minimal shuffling of information. Thus, several highspeed processors may be searching for different types of exotic behavior concurrently.

Third, multiprocessor systems hold the possibility of being the vehicle for implementing processing systems which are fault tolerant. This results because microprocessors are sufficiently low priced to allow redundancy of subsystems. This redundancy also permits parallel execution of code for software fault checking. Items to be studied include methods of fault detection, modes of system recovery, and methods of automatic system reconfiguration.

APPENDIX A

PREPROCESSOR SUPPORT ALGORITHMS

The algorithms appearing on the following pages are executed by the communication microprocessor of MAP to support the preprocessor tracking function which is executed by preprocessor firmware. The tracking and support algorithms execute concurrently with the support algorithms assuming a background role to the non-interruptable tracking function. The names of the support algorithms and their functions are given below followed by their flowcharts.

Preprocessor Initialization (PPINIT) - This routine is initiated during the power up sequence and prepares the preprocessor for its tracking operation. Its main function is to establish initial parameter limits and select the mode of operation. The end of the initialization phase contains an idle loop which is exited by an interrupt request for service. Upon completion of any service operation the idle routine is re-entered.

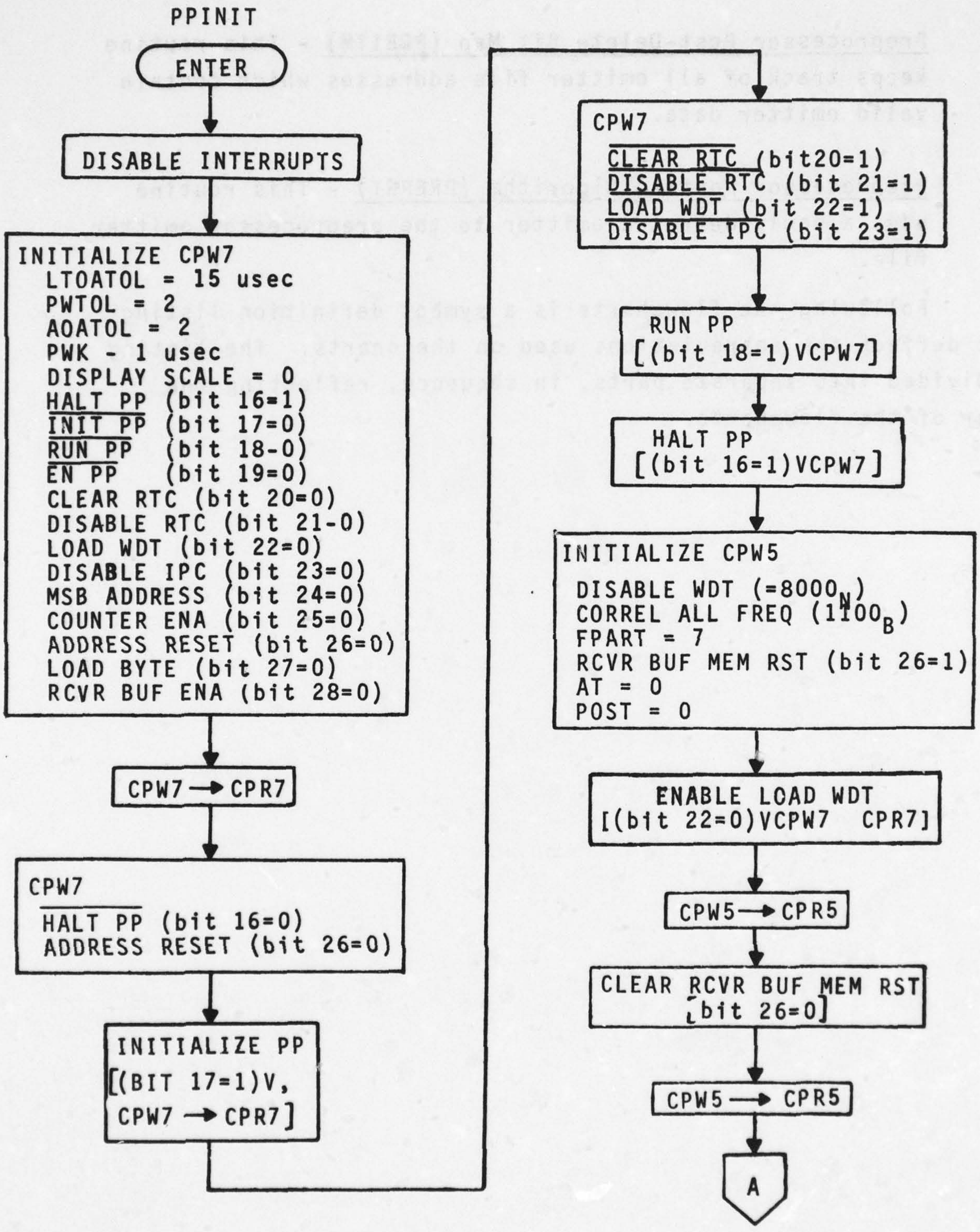
Preprocessor Clear Emitter File Memory (PCLREF) - This routine searches the emitter file for old emitters which are no longer being tracked by the preprocessor. This routine also removes the old emitters from the preprocessor file.

Preprocessor Watchdog Timer Service (PPWDTS) - This routine establishes the basic iteration rates for all service routines.

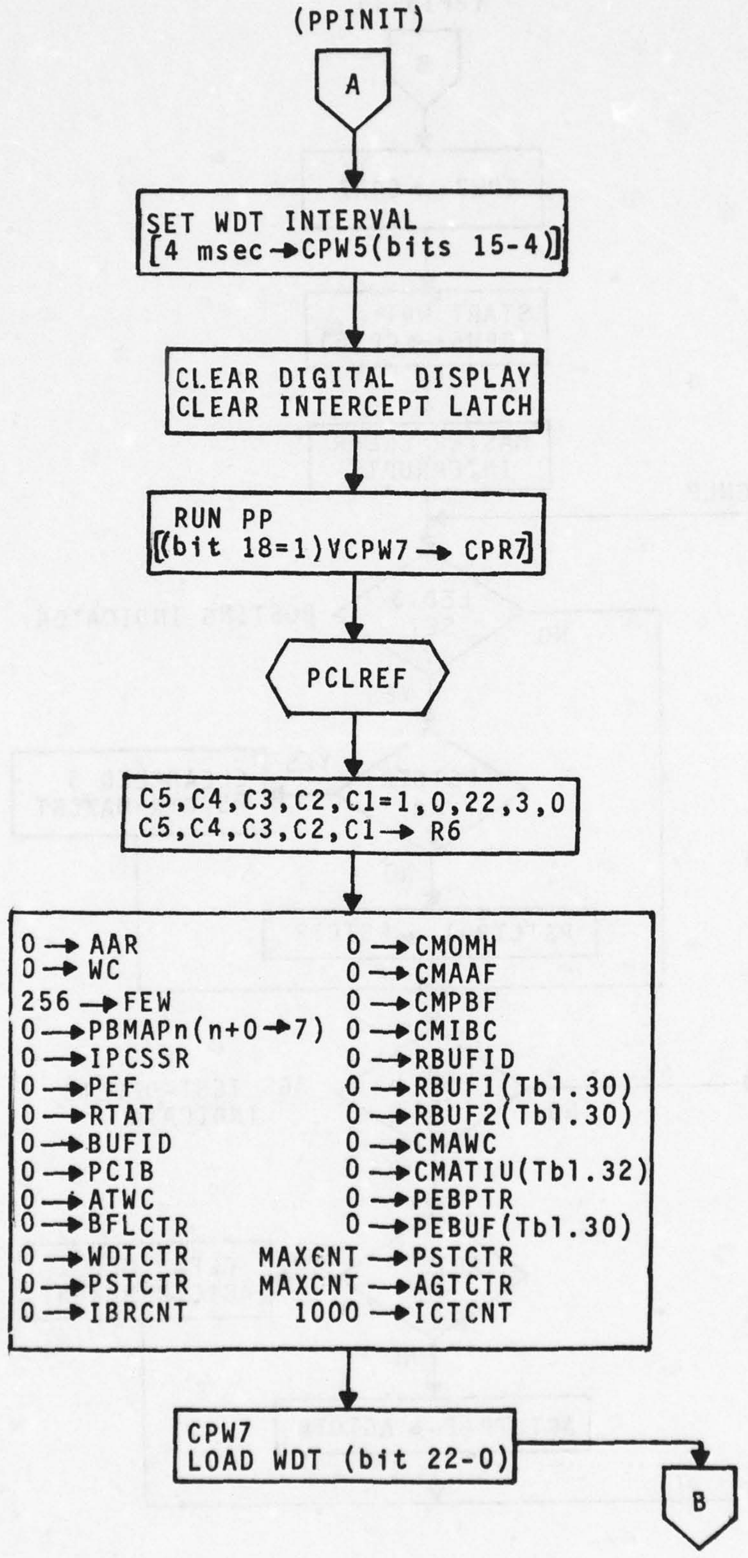
Preprocessor Post-Delete Bit Map (PDBITM) - This routine keeps track of all emitter file addresses which contain valid emitter data.

Preprocessor Posting Algorithm (PREPST) - This routine adds a newly detected emitter to the preprocessor emitter file.

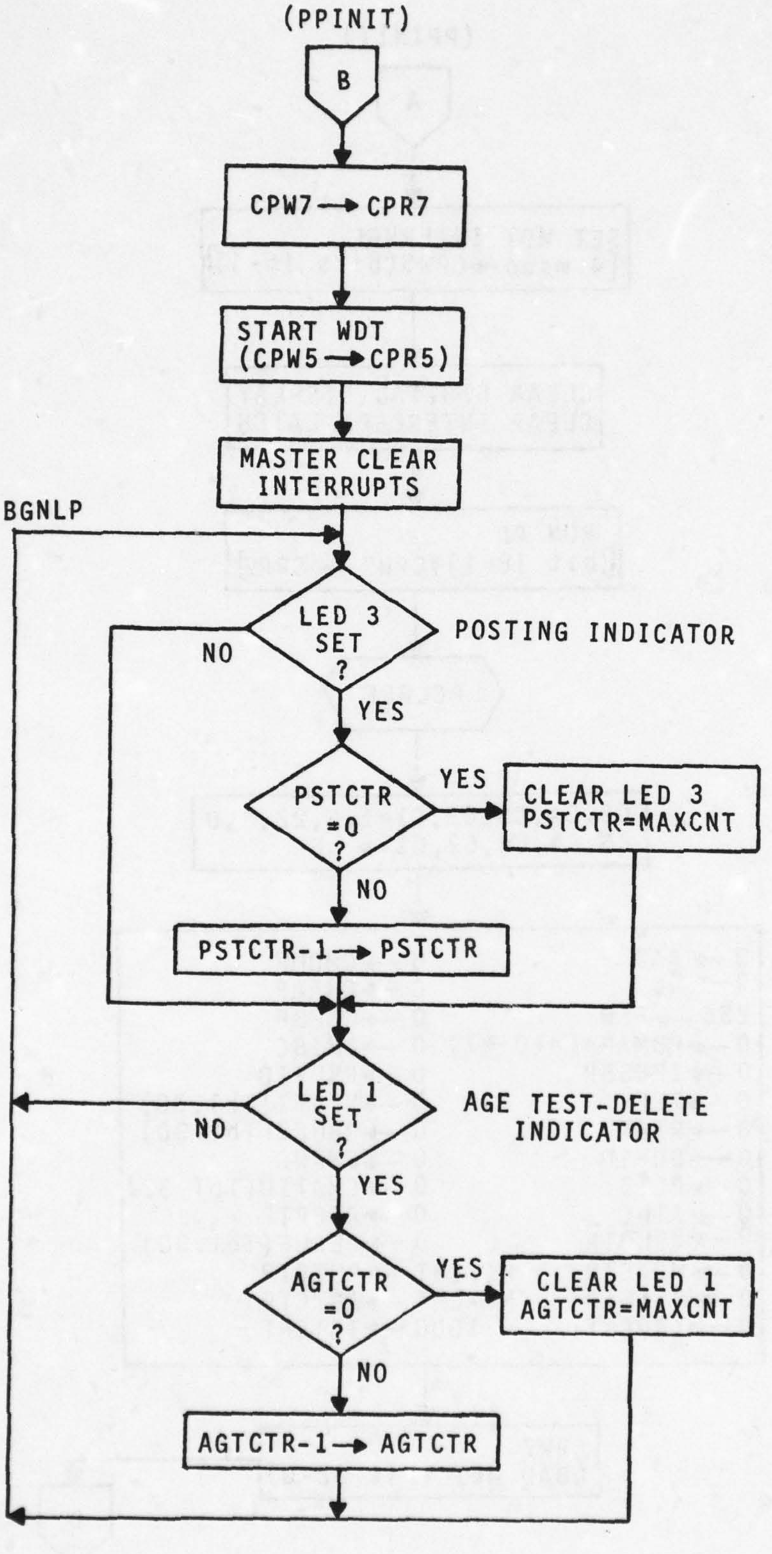
Following the flowcharts is a symbol definition listing that defines the abbreviations used on the charts. The listing is divided into separate parts, in sequence, reflecting the order of the flowcharts.



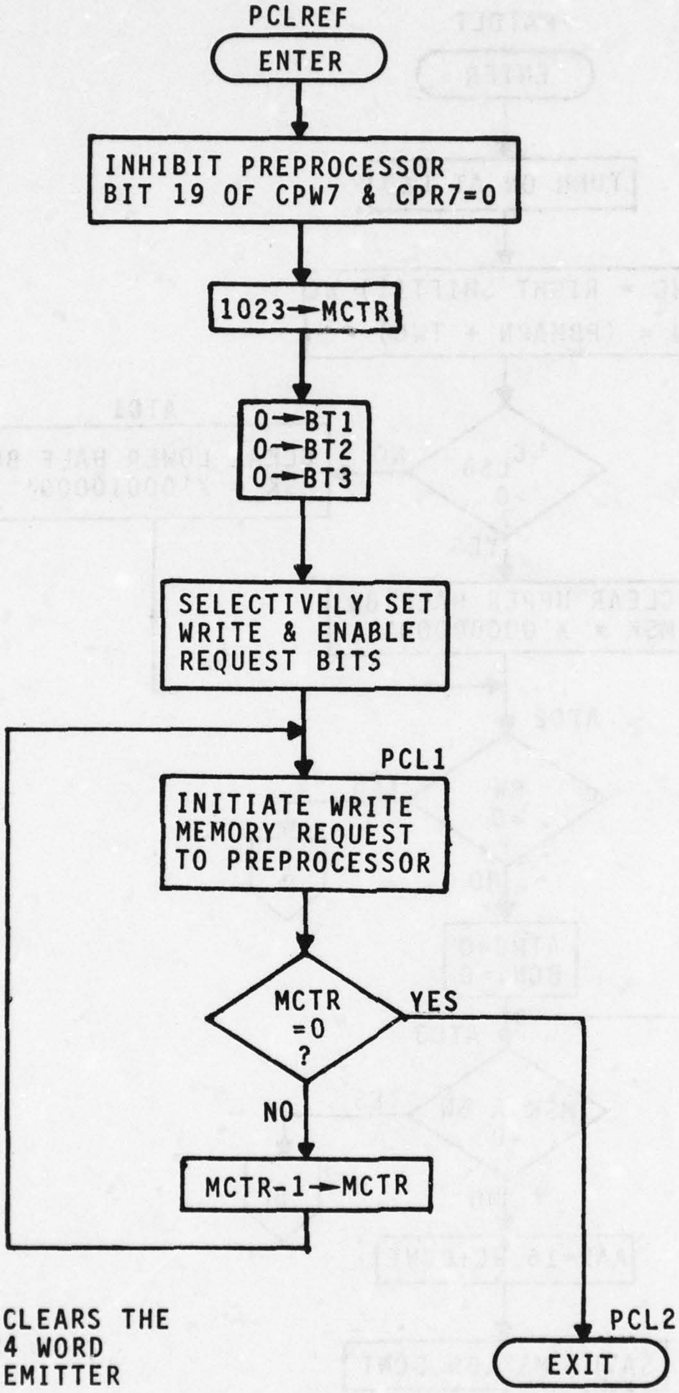
PREPROCESSOR INITIALIZATION (Sheet 1 of 3)



PREPROCESSOR INITIALIZATION (Sheet 2 of 3)

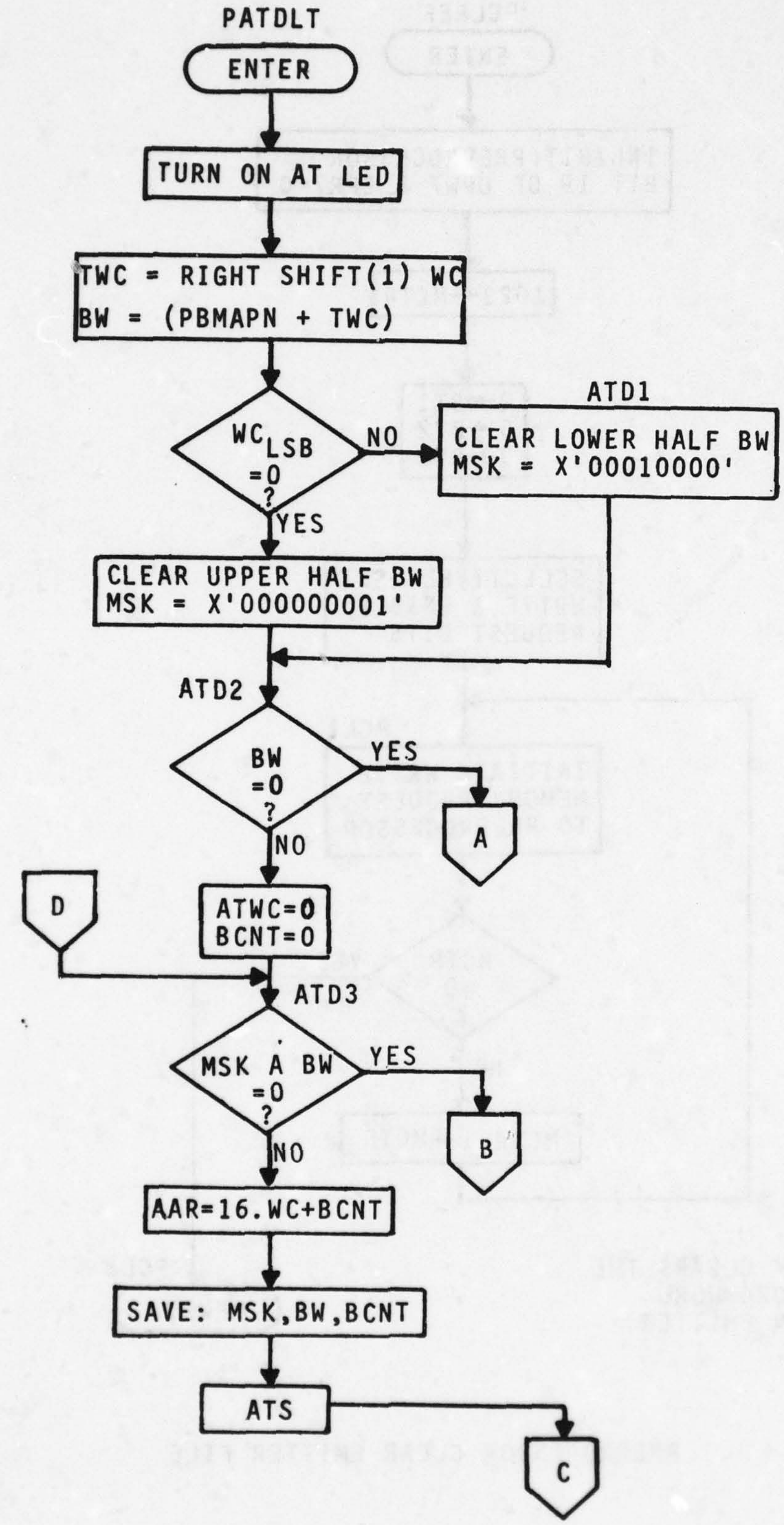


PREPROCESSOR INITIALIZATION (Sheet 3 of 3)

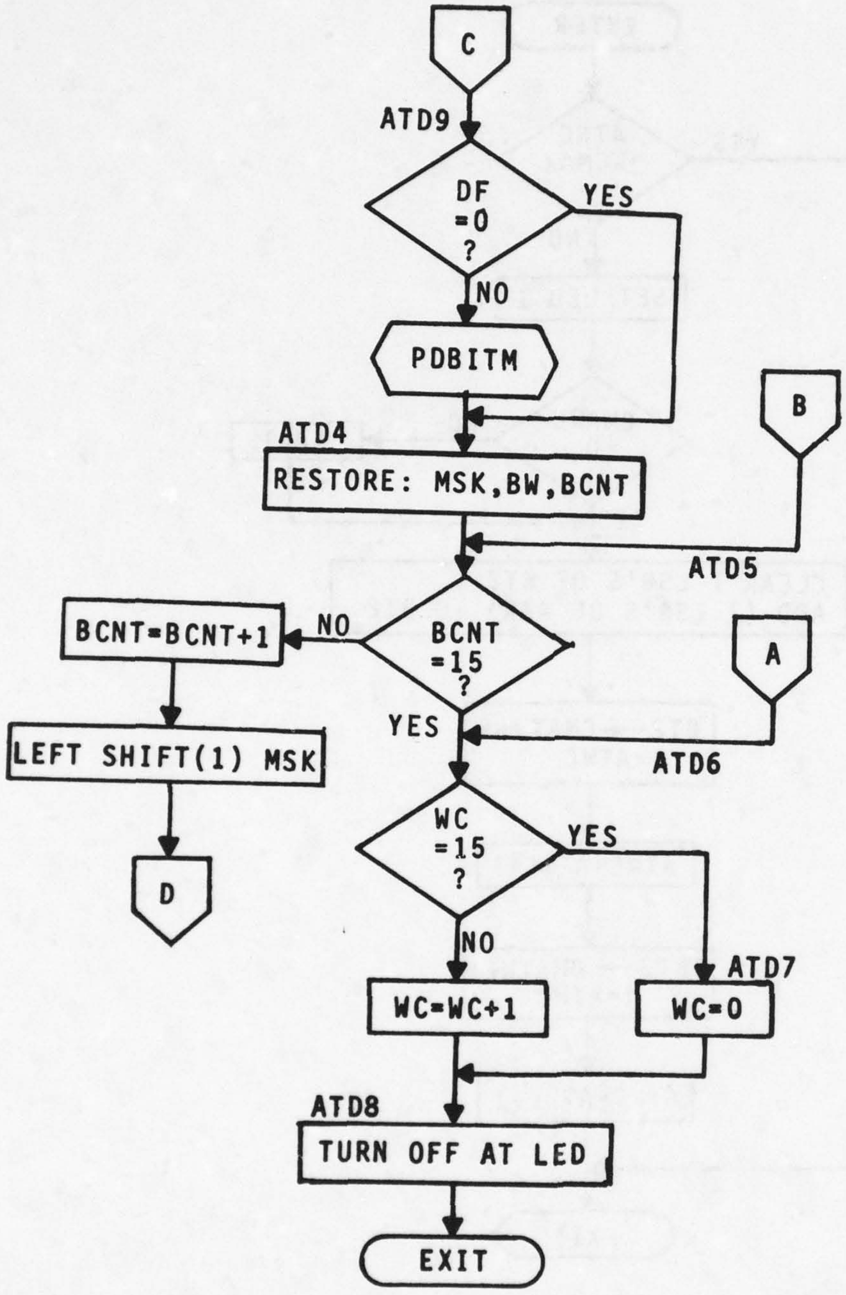

PCLREF
ENTER
INHIBIT PREPROCESSOR BIT 19 OF CPW7 & CPR7=0
1023→MCTR
0→BT1
0→BT2
0→BT3
SELECTIVELY SET WRITE & ENABLE REQUEST BITS
PCL1
INITIATE WRITE MEMORY REQUEST TO PREPROCESSOR
MCTR =0 ?
YES
NO
MCTR-1→MCTR
PCL2
EXIT

SUBROUTINE-
SEQUENTIALLY CLEARS THE 88 BIT BY 1024 WORD PREPROCESSOR EMITTER FILE MEMORY.

PREPROCESSOR CLEAR EMITTER FILE

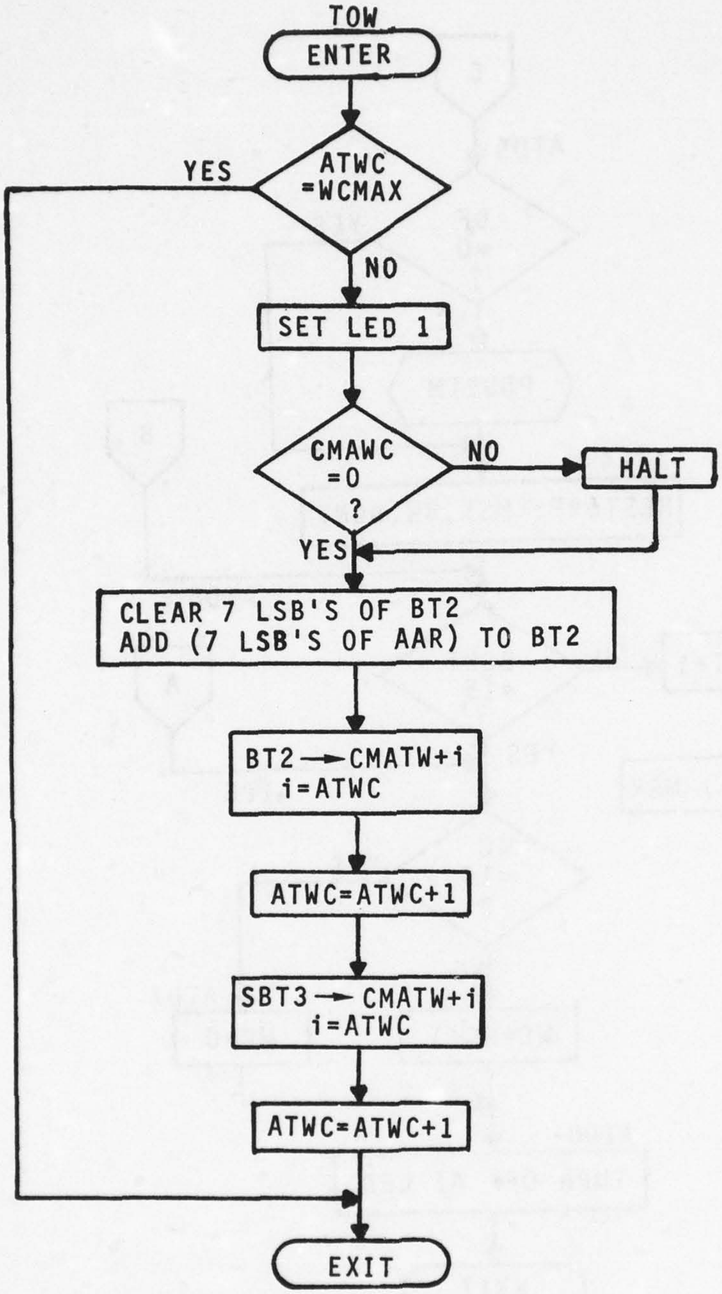


PREPROCESSOR AGE TEST DELETE (Sheet 1 of 2)

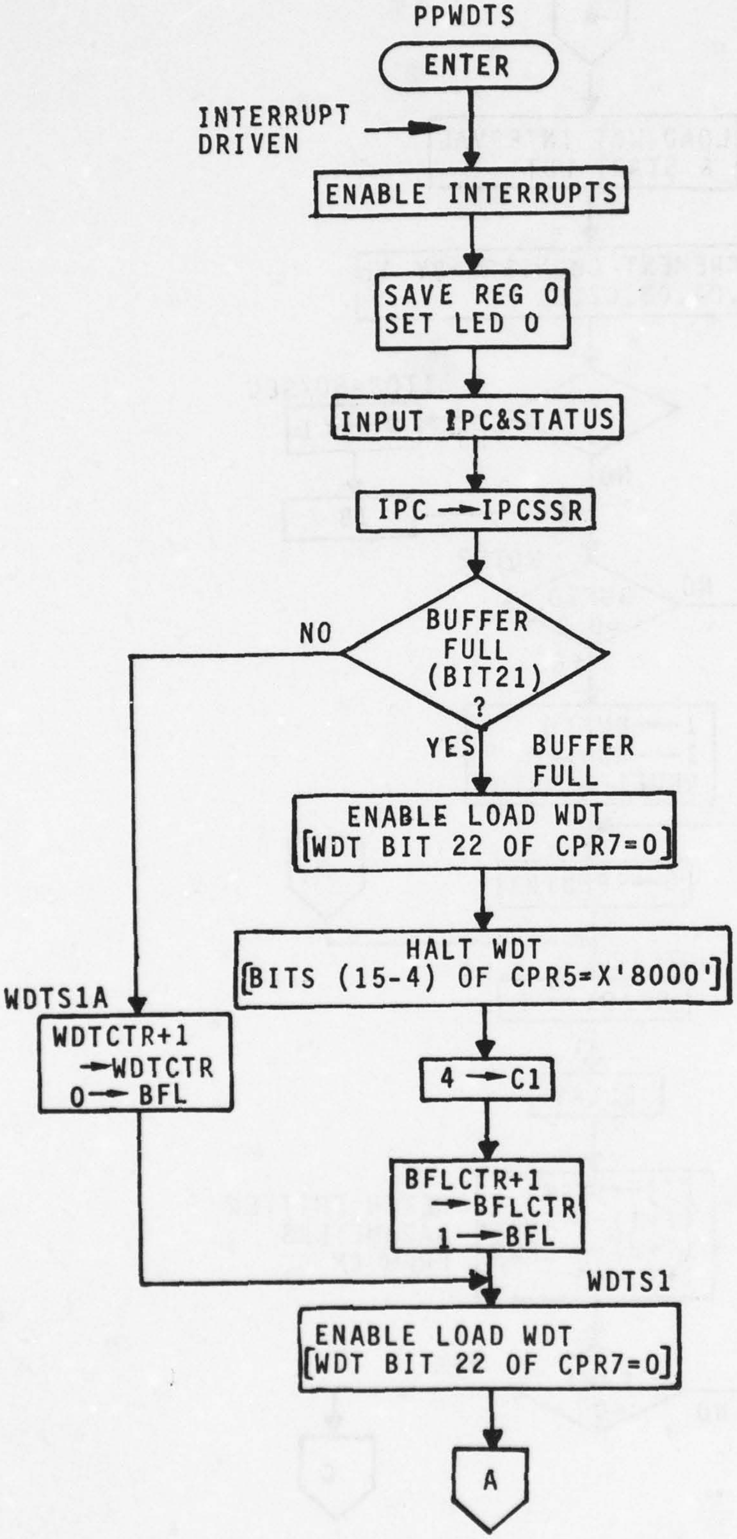


PREPROCESSOR AGE TEST DELETE (Sheet 2 of 2)

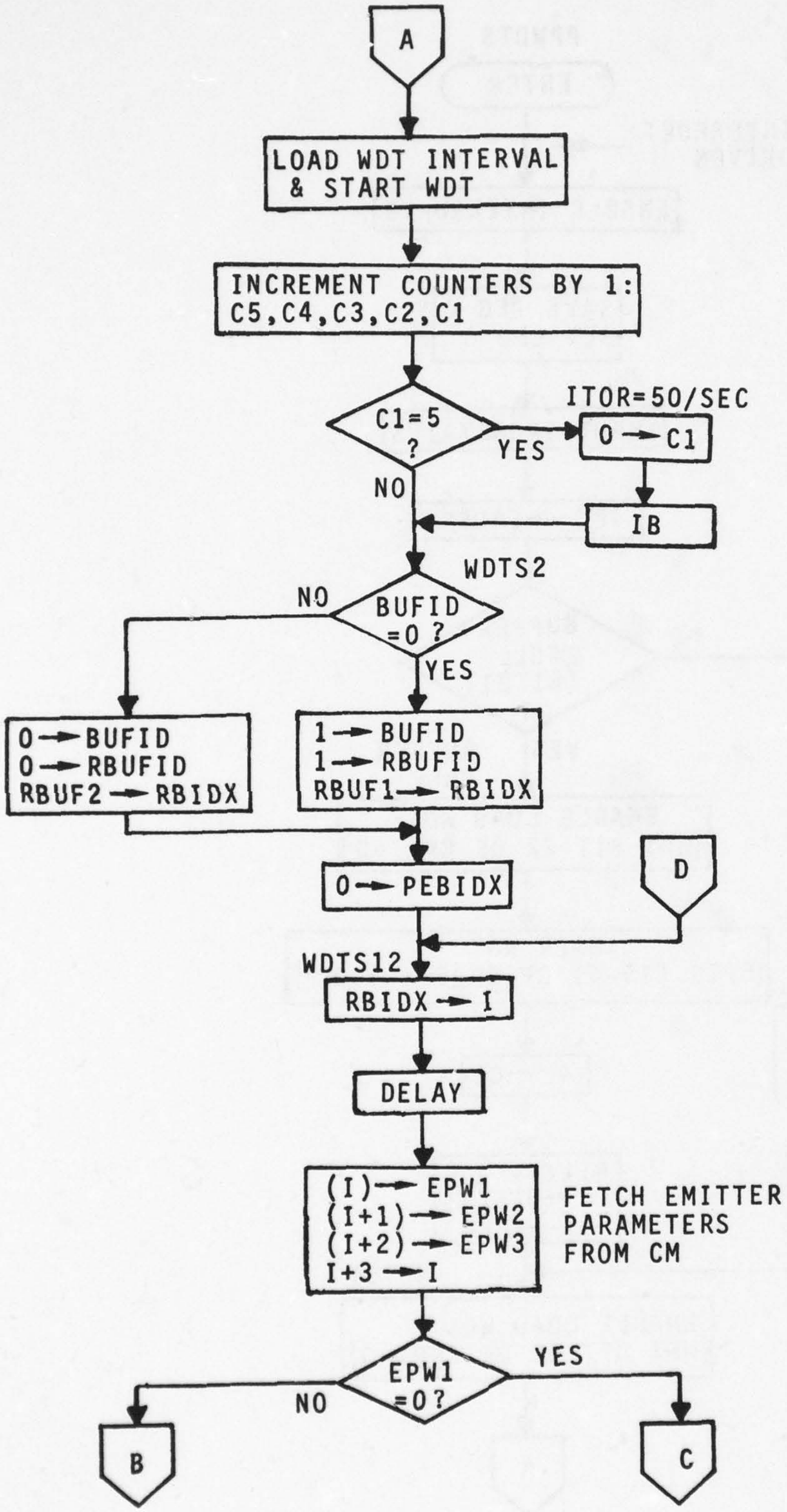
PATDLT



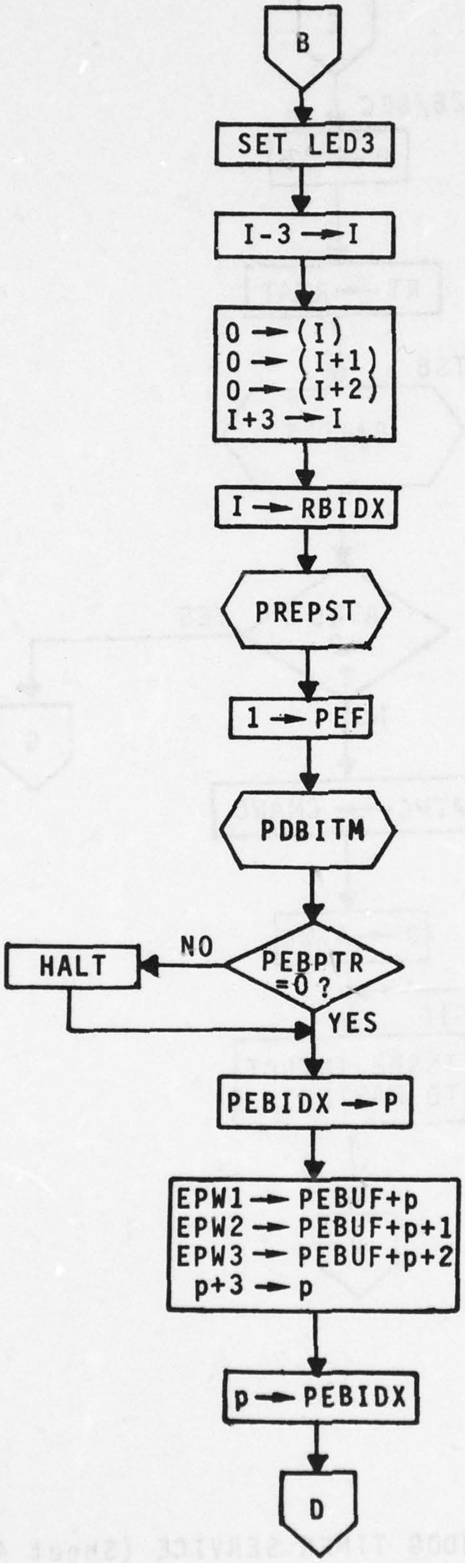
TRANSFER OLD WORD SUB PROGRAM OF
PREPROCESSOR AGE TEST DELETE ROUTINE



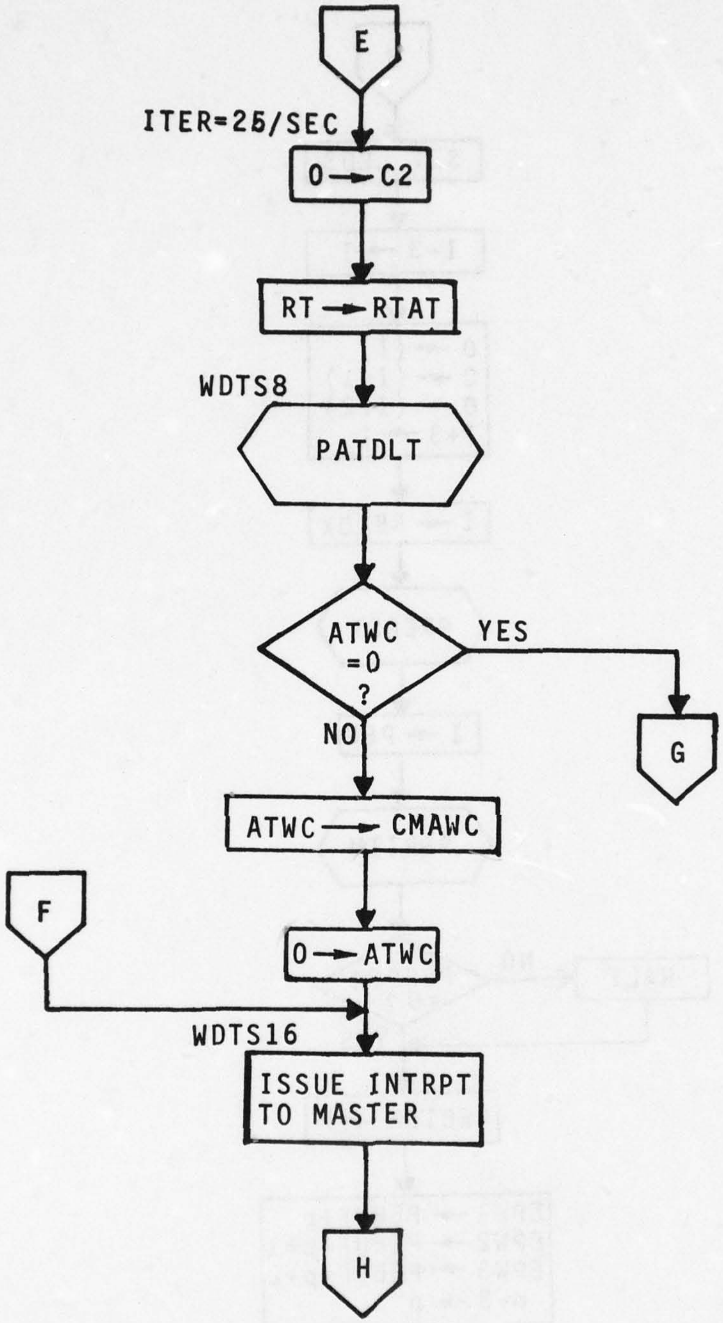
PREPROCESSOR WATCHDOG TIMER SERVICE (Sheet 1 of 5)



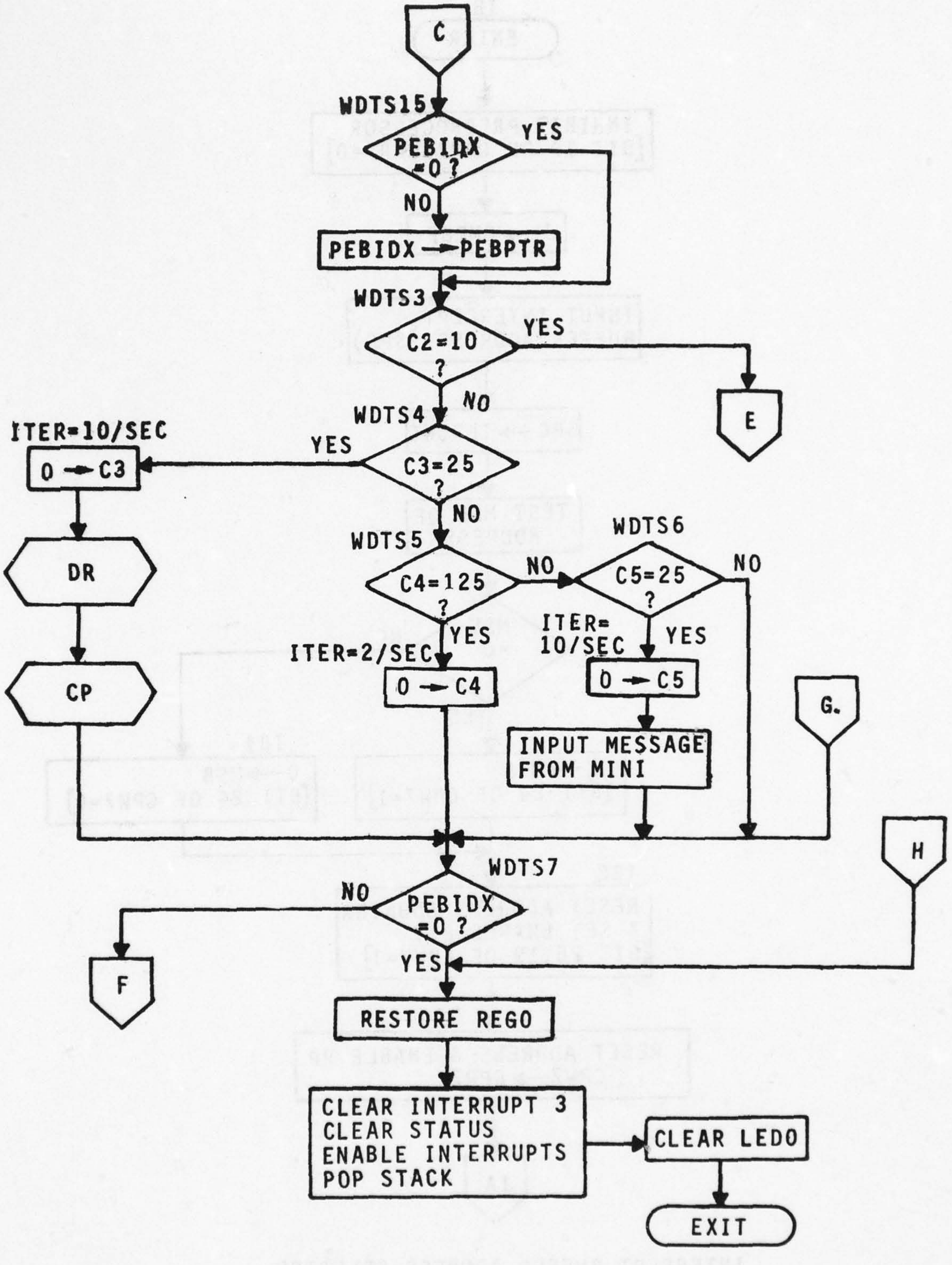
PREPROCESSOR WATCHDOG TIMER SERVICE (Sheet 2 of 5)



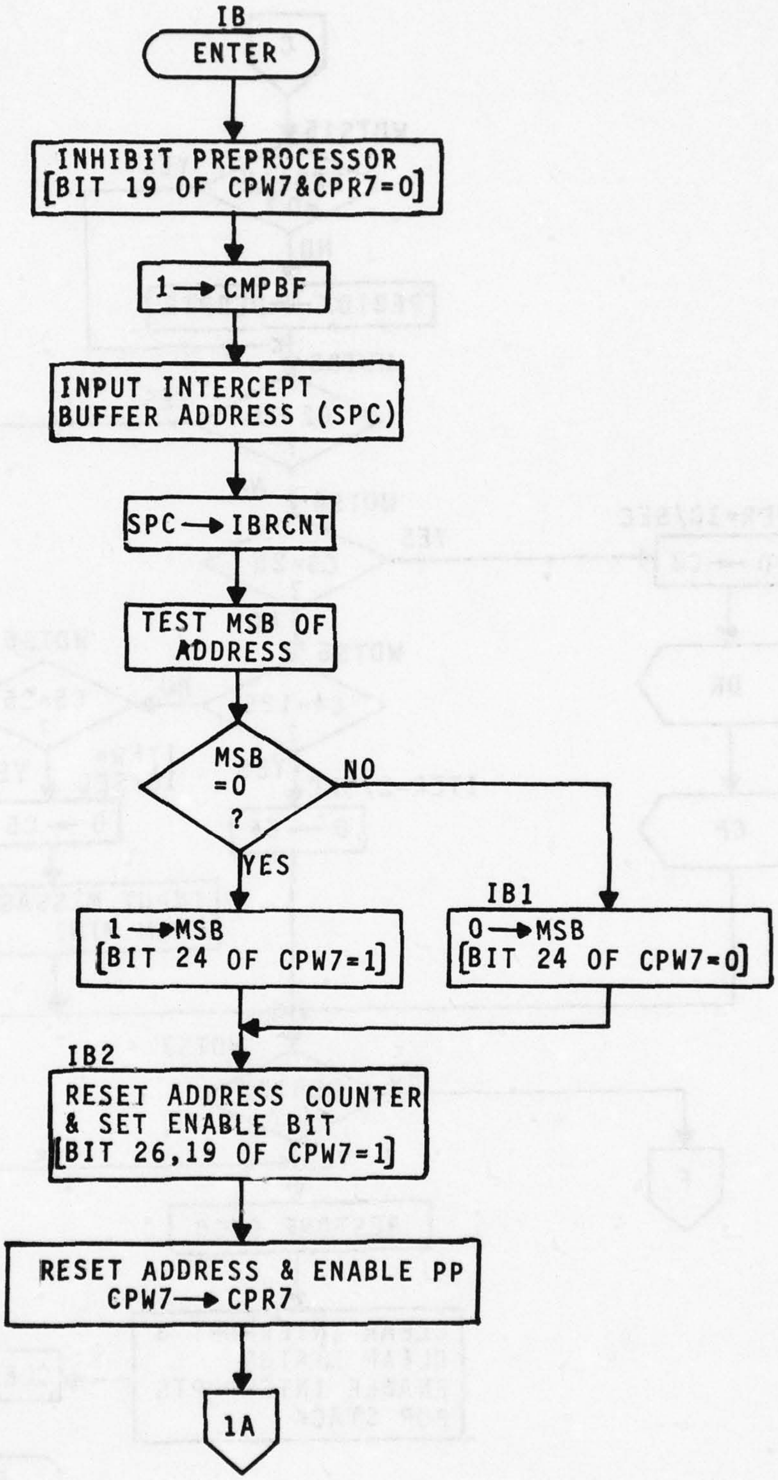
PREPROCESSOR WATCHDOG TIMER SERVICE (Sheet 3 of 5)



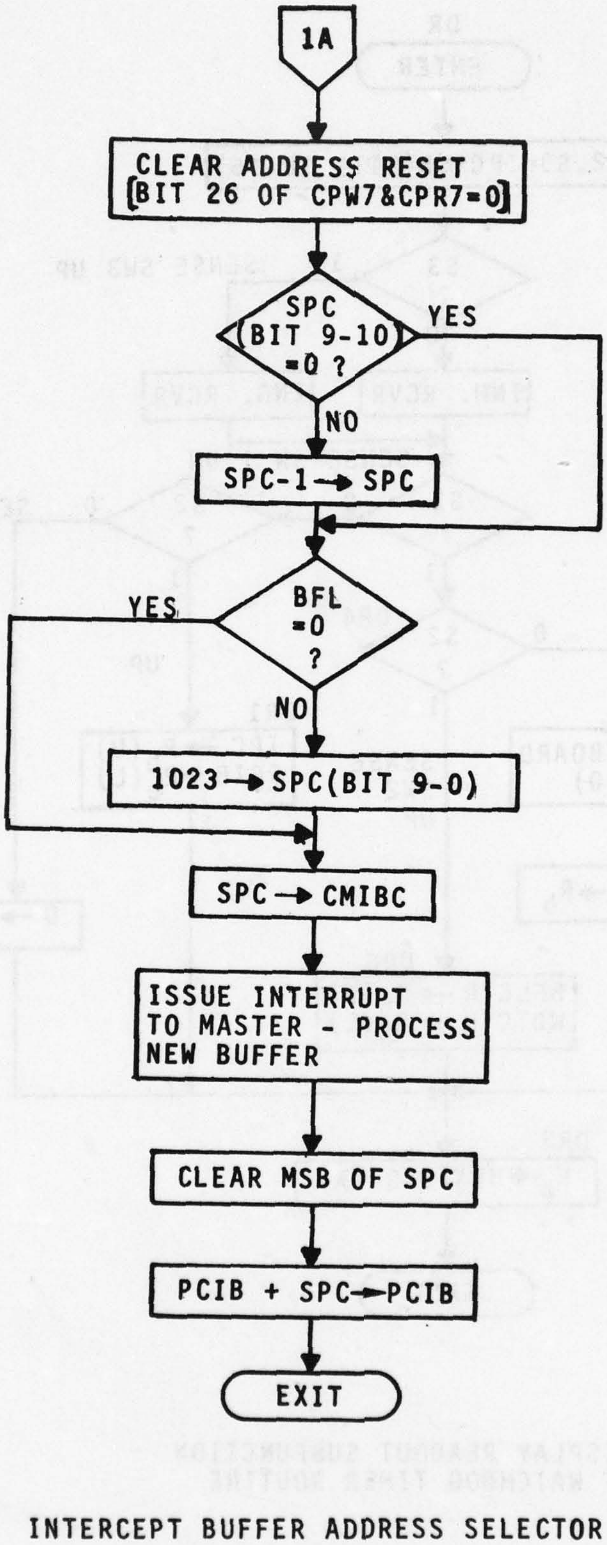
PREPROCESSOR WATCHDOG TIMER SERVICE (Sheet 4 of 5)



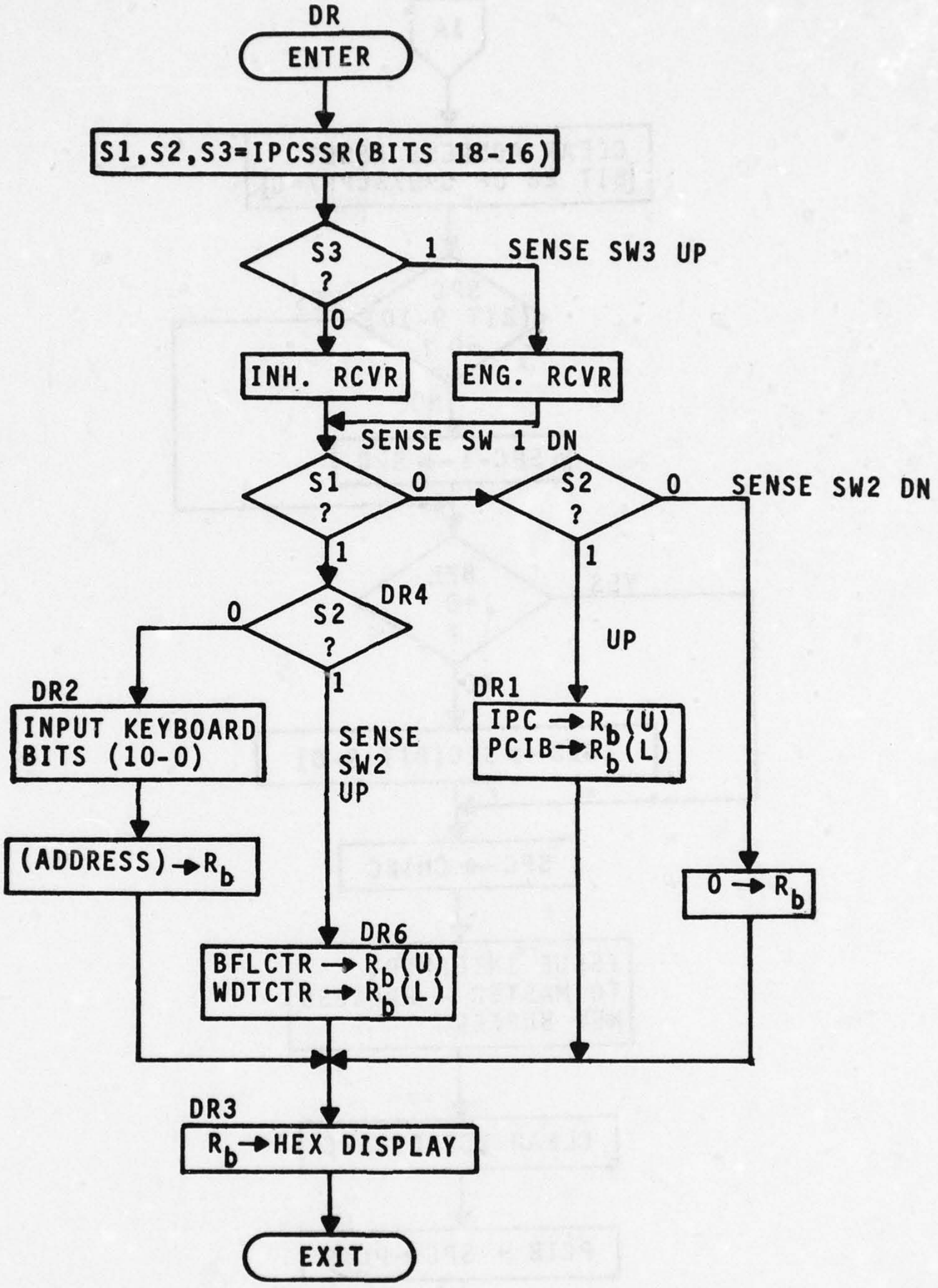
PREPROCESSOR WATCHDOG TIMER SERVICE (Sheet 5 of 5)


IB
ENTER
INHIBIT PREPROCESSOR [BIT 19 OF CPW7&CPR7=0]
1 → CMPBF
INPUT INTERCEPT BUFFER ADDRESS (SPC)
SPC → IBRCNT
TEST MSB OF ADDRESS
MSB =0 ?
NO
YES
IB1
1 → MSB [BIT 24 OF CPW7=1]
0 → MSB [BIT 24 OF CPW7=0]
IB2
RESET ADDRESS COUNTER & SET ENABLE BIT [BIT 26,19 OF CPW7=1]
RESET ADDRESS & ENABLE PP CPW7 → CPR7
1A

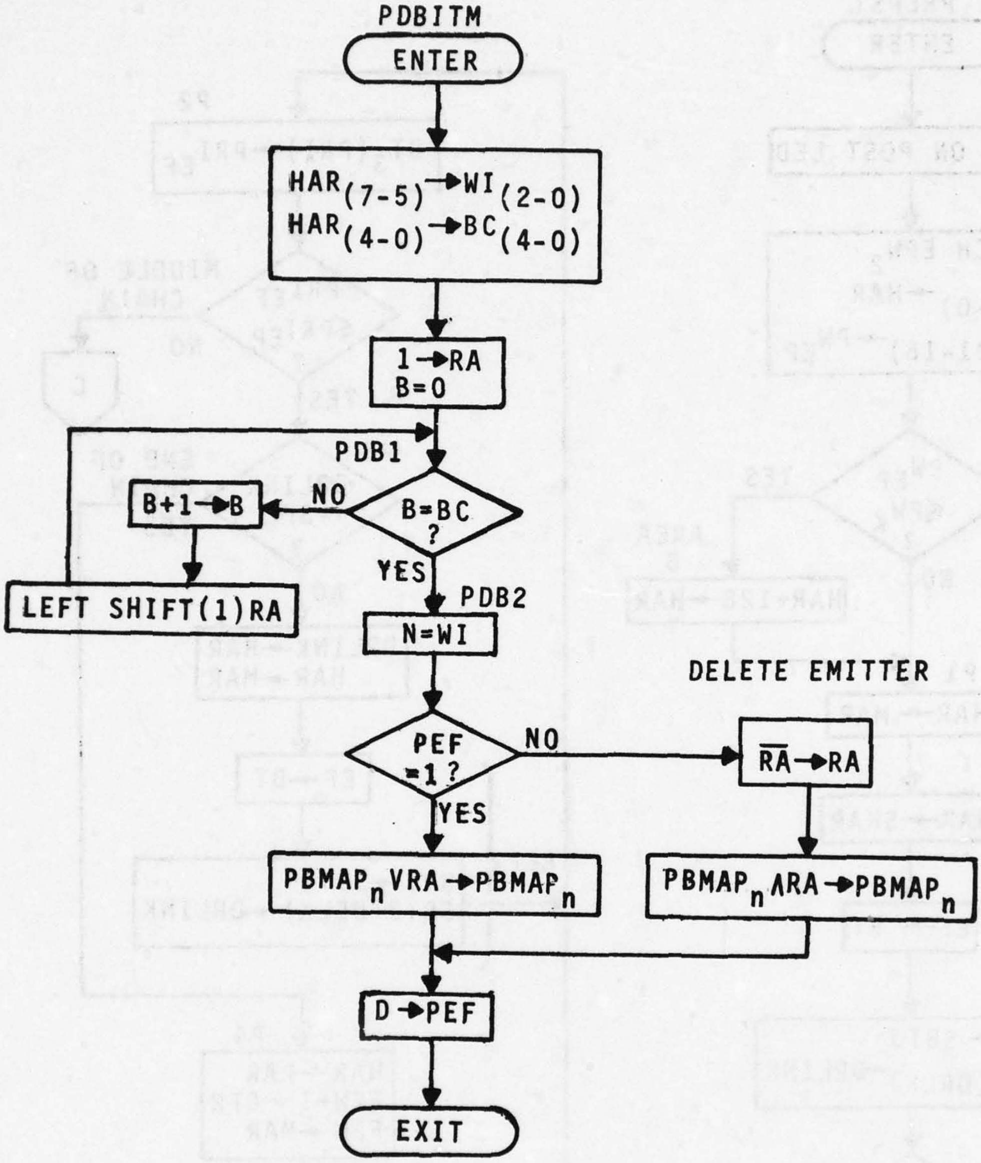
INTERCEPT BUFFER ADDRESS SELECTOR
SUBFUNCTION OF
WATCHDOG TIMER ROUTINE (Sheet 1 of 2)



INTERCEPT BUFFER ADDRESS SELECTOR
SUBFUNCTION OF
WATCHDOG TIMER ROUTINE (Sheet 2 of 2)

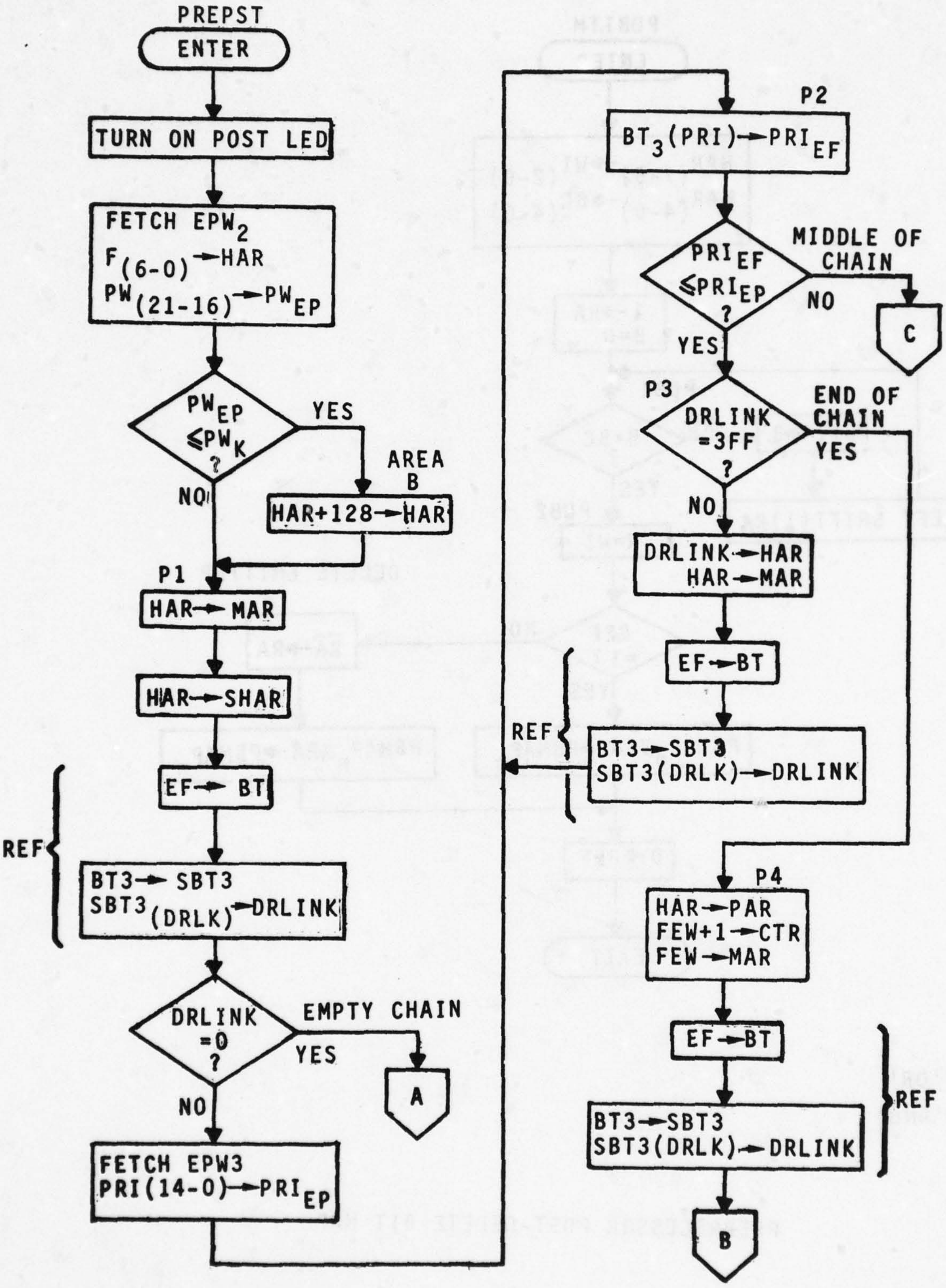


DISPLAY READOUT SUBFUNCTION
OF WATCHDOG TIMER ROUTINE

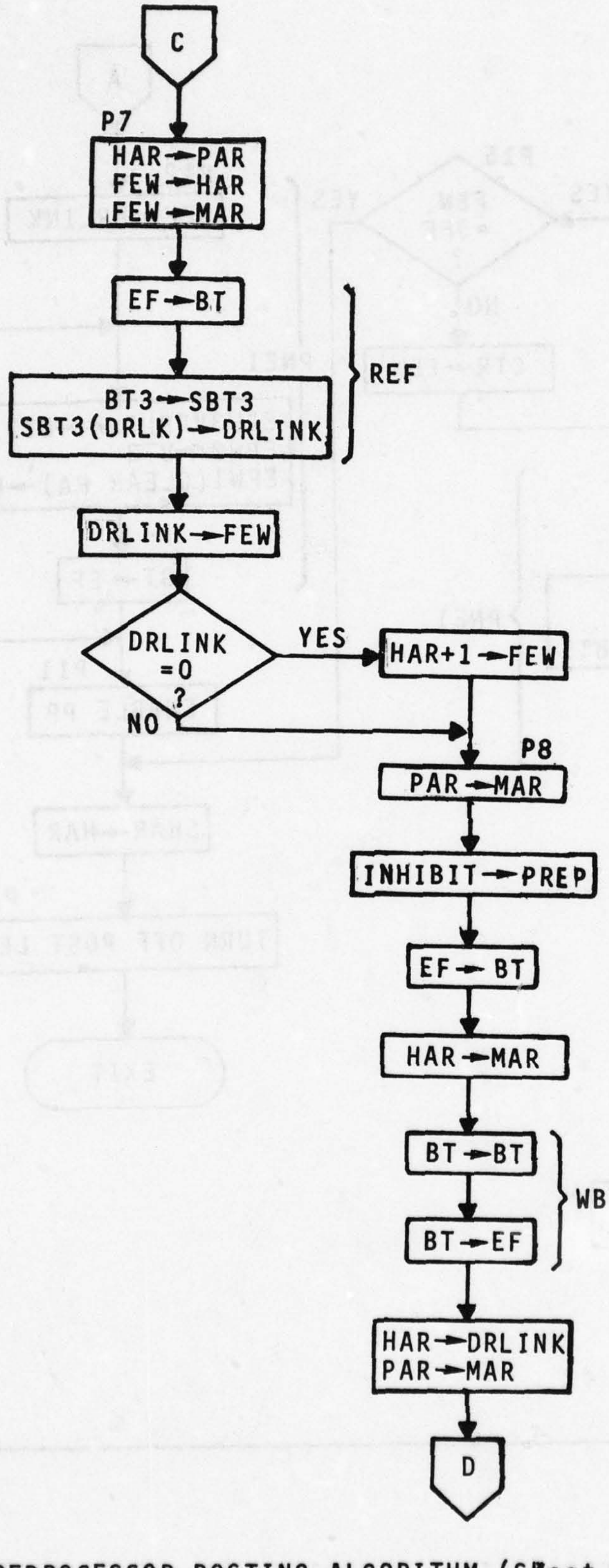


V='OR'
Λ='AND'

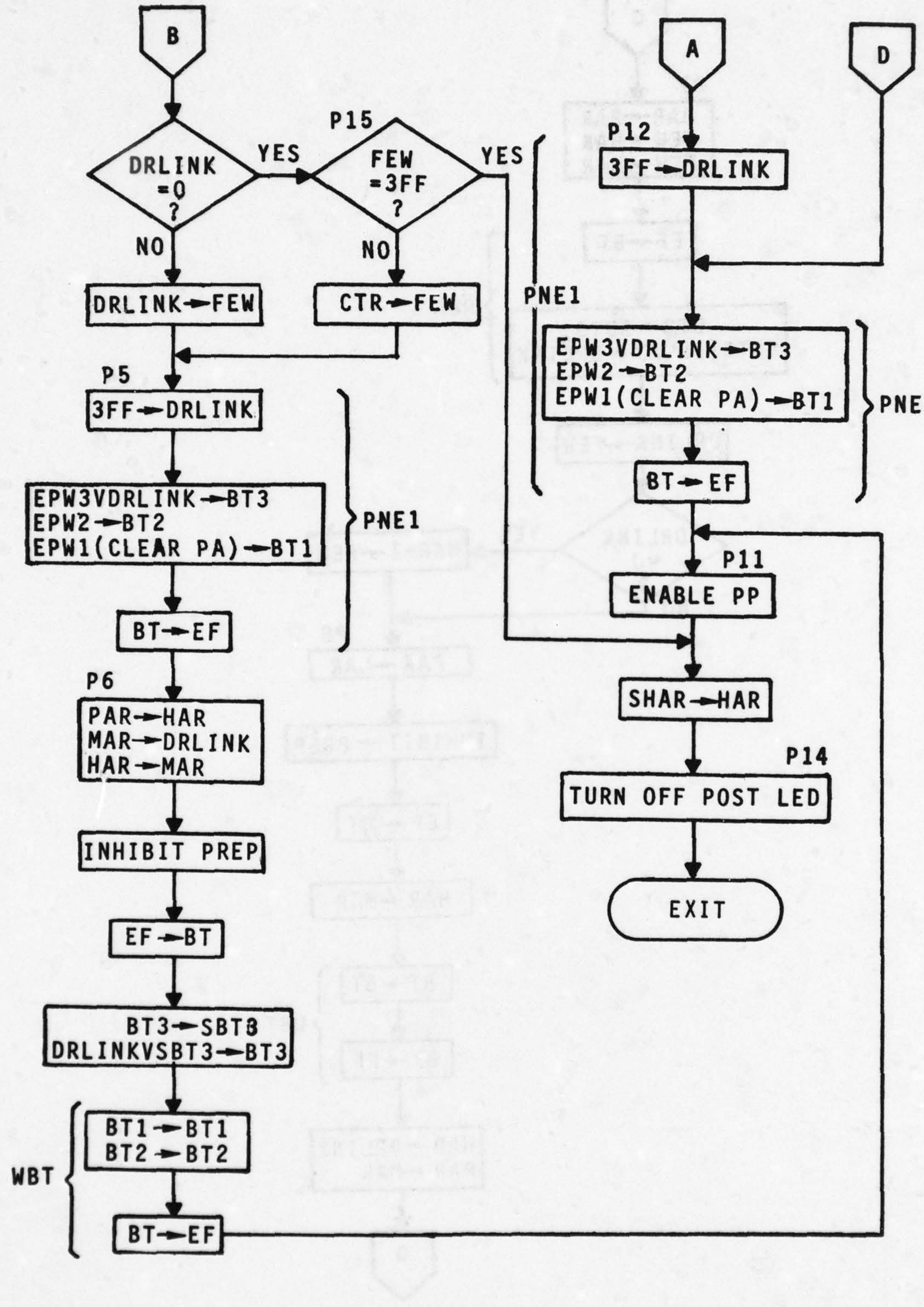
PREPROCESSOR POST-DELETE BIT MAP



PREPROCESSOR POSTING ALGORITHM (Sheet 1 of 3)



PREPROCESSOR POSTING ALGORITHM (SHeet 2 of 3)



PREPROCESSOR POSTING ALGORITHM (Sheet 3 of 3)

SYMBOL DEFINITIONS

PPINIT

AAR	AGE TEST ADDRESS REGISTER
CPR5	PREPROCESSOR COMMAND-PARAMETER REG. 5 DEVICE ADDRESS
CPR7	PREPROCESSOR COMMAND-PARAMETER REG. 7 DEVICE ADDRESS
CPW5	COMMAND-PARAMETER WORD 5
CPW7	COMMAND-PARAMETER WORD 7
DDRO	PREPROCESSOR DIGITAL DISPLAY READOUT DEVICE ADDRESS
FEW	FIRST EMPTY WORD
INTL	PREPROCESSOR INTERCEPT LATCH DEVICE ADDRESS
PBMAPN	POST BIT MAP WORD - RESERVE 8
PCIB	PULSE COUNT, INTERCEPT BUFFER
PEF	POSTING EMITTER FLAG
PPTR	POSTING POINTER
RTAT	REAL TIME, AGE TEST
IPCSSR	INP. PLS. CNTR., SENSE, STATUS
WC	WORD COUNT - AGE TEST
BUFIDQ	BUFFER IDENT.
BFLCTR	BUFFER FULL COUNTER
WDTCTR	WATCHDOG TIMEOUT COUNTER
RBUF1	REPORT BUFFER 1
RBUF2	REPORT BUFFER 2
RBUFID	REPORT BUFFER IDENT.
CMAWC	COMMON MEMORY AGE TESTED WORD COUNT
CMATW	COMMAND MEMORY AGE TESTED WORD
PEBPTR	POST EMITTER BUFFER POINTER
PEBUF	POST EMITTER BUFFER

PDBITM

BC	BIT COUNT
BCNT	COUNTER
HAR	HASH ADDRESS REGISTER
PBMAPN	POST BIT MAP WORD-RESERVE 8
PEF	POSTING EMITTER FLAG
WI	WORD INDEX NUMBER

PPWDTS

C1	ITERATION COUNTER 1
C2	" " 2
C3	" " 3
C4	" " 4
C5	" " 5
EPCM1	EMITTER PARAM. COMM. MEMORY 1
EPCM2	" " " " 2
EPCM3	" " " " 3

SYMBOL DEFINITIONS

<u>PPWDTS</u> (cont'd)

CMOMH	COMM. MEMORY OUTPUT MESS. HEADER ADDRESS
CPR5	PREPROCESSOR COMMAND-PARAMETER REG 5 DEVICE ADDRESS
CPR7	PREPROCESSOR COMMAND-PARAMETER REG 7 DEVICE ADDRESS
CPW5	COMMAND PARAMETER WORD 5
CPW7	COMMAND PARAMETER WORD 7
DDRO	PREPROCESSOR DIGITAL DISPLAY READOUT
IBAR	PREPROCESSOR INTERCEPT BUF. ADRS. DEVICE ADDRESS
IBFUL	INTERCEPT BUF. FUL FLAG
IPCSR	PREPROCESSOR INP PLS CNTR & STA DEVICE ADDRESS
KBR	PREPROCESSOR KEYBOARD REG. DEVICE ADDRESS
EPW1	EMITTER PARAMETER WORD 1
EPW2	" " " 2
EPW3	" " " 3
PCIB	PULSE COUNT, INTERCEPT BUFFER
PEF	POSTING EMITTER FLAG
PPTR	POSTING POINTER
RTAT	REAL TIME AGE TEST
RTCR	PREPROCESSOR REAL TIME CLOCK REG. DEVICE ADDRESS
SPC	SAVE PULSE COUNT
MHDG	MSSG HDR
ATWC	AGE TESTED WORD COUNTER
PEBPTR	POST EMITTER BUFFER POINTER
PEBUF	POST EMITTER BUFFER
CMAWC	COMMON MEMORY AGE TESTED WORD COUNT
CMPBF	COMMON MEMORY PROCESS BUFFER FLAG
CMIBC	COMMON MEMORY INTERCEPT BUFFER COUNT
RBUFID	REPORT BUFFER IDENT.
RBUF1	REPORT BUFFER 1
RBUF2	REPORT BUFFER 2

<u>PCLREF</u>

BT1	PREPROCESSOR BUS TRANCIEVER 1 DEVICE ADDRESS
BT2	" " " 2 " "
BT3	" " " 3 " "
EFAR	PREPROCESSOR EMITTER FILE ADRS DEVICE ADDRESS
CPR7	PREPROCESSOR COMMAND PARAMETER REG. 7 DEVICE ADDRESS
MCTR	MEMORY ADDRESS COUNTER

<u>PREPST</u>

BT	PREPROCESSOR BUS TRANSCEIVERS (BT1, BT2, BT3)
BT1	BUS TRANSCEIVER 1
BT2	" " 2
BT3	" " 3
CPR5	PREPROCESSOR COMMAND-PARAMETER REG. 5 DEVICE ADDRESS
CPR7	PREPROCESSOR COMMAND-PARAMETER REG. 7 DEVICE ADDRESS
CPW5	COMMAND PARAMETER WORD 5
CPW7	COMMAND PARAMETER WORD 7

SYMBOL DEFINITIONS

PREPST (cont'd)

CTR	COUNTER, TEMP
DRLINK	DATA REGISTER LINK
EF	EMITTER FILE MEMORY
EFAR	EMITTER FILE ADDRESS REGISTER
EPW1	EMITTER PARAMETER WORD 1
EPW2	" " " 2
EPW3	" " " 3
F	PULSE FREQUENCY
FEW	FIRST EMPTY WORD
HAR	HASH ADDRESS REGISTER
MAR	MEMORY ADDRESS REGISTER
PA	PULSE AMPLITUDE
PAR	PREVIOUS ADDRESS REGISTER
PRI	PULSE REPETITION INTERVAL
PRIEF	PRI - EMITTER FILE
PRIEP	PRI - EMITTER PULSE
PW	PULSE WIDTH
PWEP	PW - EMITTER PULSE
PWK	PULSE WIDTH - BOUNDARY VALUE
SBT3	SAVE BUS TRANSCEIVER 3
SHAR	SAVE HASH ADDRESS

COMMON MEMORY

RBUFID	REPORT BUFFER IDENT.
RBUF1	REPORT BUFFER 1 (RESERVE 30)
RBUF2	REPORT BUFFER 2 (RESERVE 30)
CMOMH	COMMON MEMORY OUTPUT MSSG HDR
CMAAF	C " " AGE TEST ACKNOWLEDGE FLAG
CMPBF	" " PROCESS BUFFER FLAG
CMIBC	" " INTERCEPT BUFFER COUNT
PEBPTR	POSTED EMITTER BUFFER POINTER
PEBUF	POSTED EMITTER BUFFER (RESERVE 30)
CMAWC	COMMON MEMORY AGE TESTED WORD COUNT
CMATW	COMMON MEMORY AGE TESTED WORD (RESERVE 32)

PATDLT

BW	BIT WORD
WC	WORD COUNT - AGE TEST
ATWC	AGE TEST WORD COUNTER
BCNT	COUNTER
AAR	AGE TEST ADDRESS REGISTER
DF	DELETE FLAG
HAR	HASH ADDRESS REGISTER
MAR	MEMORY ADDRESS REGISTER
SBT3	SAVE BUS TRANSCEIVER 3
SRLINK	DATA REGISTER LINK
RTAT	REAL TIME AGE TEST
AT	AGE TEST TIME

SYMBOL DEFINITIONS

PATDLT (cont'd)

BT	PREPROCESSOR BUS TRANSCEIVERS (BT1, BT2, BT3)
BT1	PREPROCESSOR BUS TRANSCEIVER 1 DEVICE ADDRESS
BT2	" " " 2 " "
BT3	" " " 3 " "
PAR	PREVIOUS ADDRESS REG.
EF	EMITTER FILE MEMORY
FEW	FIRST EMPTY WORD REGISTER
SVDLK	SAVE LINK
CMAWC	COMMON MEMORY AGE TESTED WORD COUNT
CMATW	COMMON MEMORY AGE TESTED WORD

APPENDIX B

PREPROCESSOR AND MULTIPROCESSOR COMMUNICATION STRUCTURE

The operation of the MAP system requires extensive communication between the preprocessor and the multiprocessor. Figure B-1 shows the preprocessor hardware which is accessible from the multiprocessor. Communication between the preprocessor and one of the microcomputers, in the multiprocessor, is governed by the use of read direct and write direct instructions in the microcomputer. A 32-bit interface bus permits bidirectional data transfer between one of the CPU's 16 general purpose (GP) registers and a preprocessor device specified by its device address. For a write operation, the CPU selects a GP register as the source and a preprocessor device as the destination. For a read operation, the CPU selects a preprocessor device as the source of data for a GP register. Table I presents a list of preprocessor devices and their respective device addresses, and references to other tables in this appendix which give detailed word formats for each device.

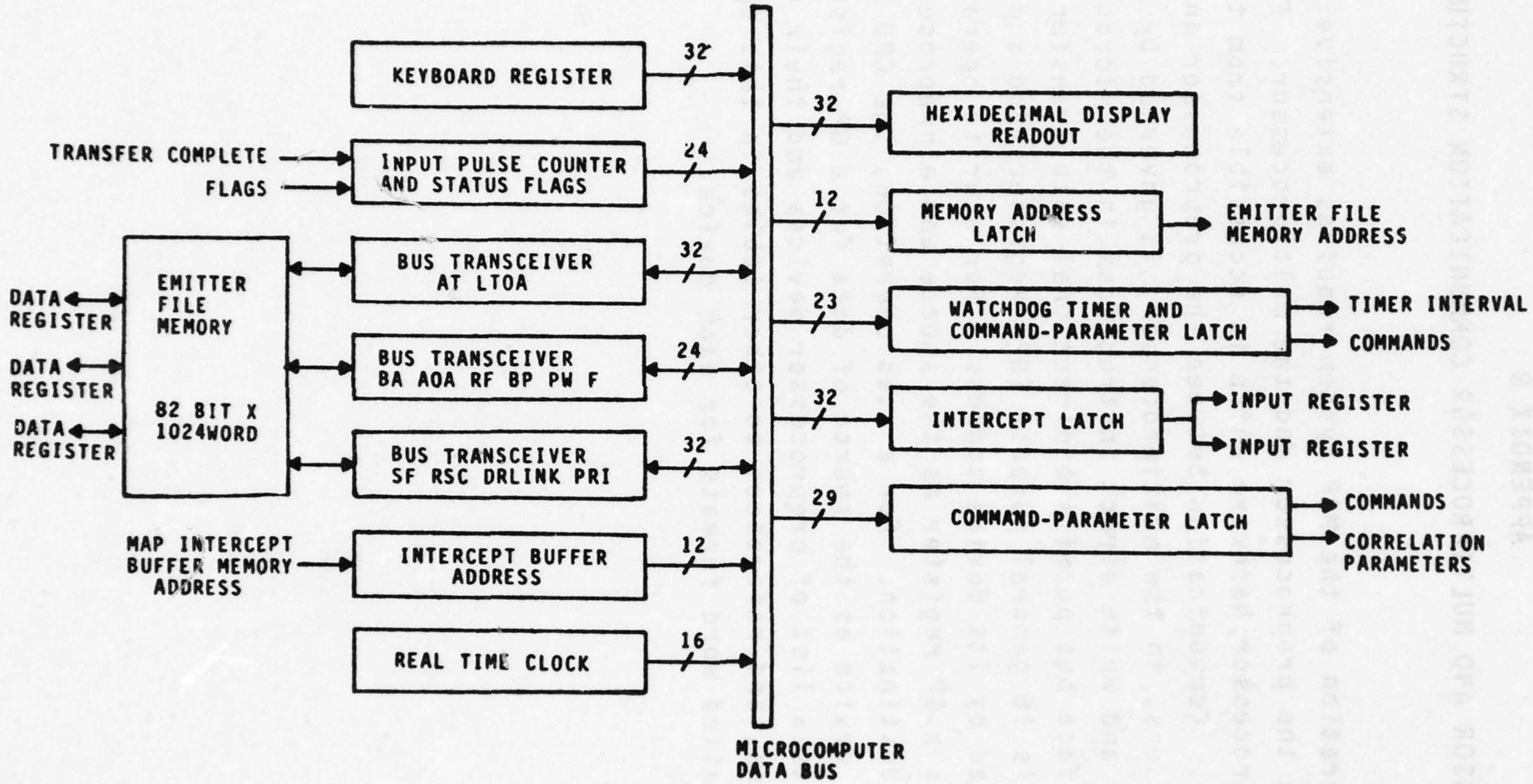


Figure B-1 - Preprocessor/Microcomputer Communication Bus

Table B-I

Preprocessor Devices

DEVICE	CPU OPERATION	MNEMONIC (HEXIDEC)	TABLE
Digital Display Readout	Write	DDRO (X'10")	B-II
Bus Transceiver Latch 1-AT,LTOA	Write/read	BT1 (X'11')	B-III
Bus Transceiver Latch 2-AOA, PW, F, flags	Write/read	BT2 (X'12')	B-IV
Bus Transceiver Latch 3-RSC, DRLINK, PR1	Write/read	BT3 (X'13')	B-V
Emitter File Address Register	Write	EFAR (X'14')	B-VI
Command-Parameter & WDT Latch	Write	CPR5 (X'15')	B-VII
Intercept Latch	Write	INTL (X'16')	B-VIII
Command-Parameter Latch	Write	CPR7 (X'17')	B-IX
Keyboard Register (Note 1.)	Read	KBR (X'10')	B-X
Input Pulse Counter & Status Reg. (Note 1.)	Read	IPCSR (X'10')	B-XI
Real Time Clock Register	Read	RTCR (X'15')	B-XII
Intercept Buffer Address	Read	IBAR (X'14')	B-XIII

TABLE B-II

DIGITAL DISPLAY READOUT

	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
DDRO(X'10')	HEX7	HEX6	HEX5	HEX4	HEX3	HEX2	HEX1	HEXØ
	MSB							LSB

TABLE B-III

BUS TRANSCEIVER LATCH 1

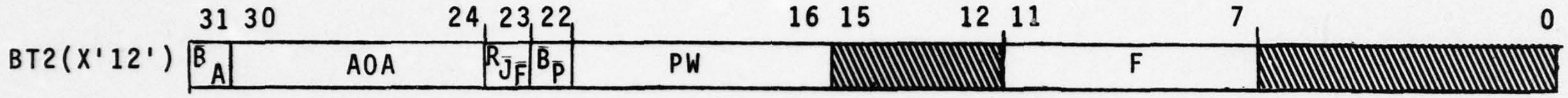
	31 … 24	23 … 0
BT1(X'11')	AT	LTOA

AT - Age Test Time

LTOA - Last Time of Arrival

TABLE B-IV

BUS TRANSCEIVER LATCH 2



31
30
24
23
22
16
15
12
11
7
0
BT2(X'12')
$\overline{\text{B}}_{\text{A}}$
AOA
$\overline{\text{R}}_{\text{JF}}$
$\overline{\text{B}}_{\text{P}}$
PW
F

BA - Bypass AOA Correlation

AOA - Angle of Arrival

RJF - Reject Pulse Intercept

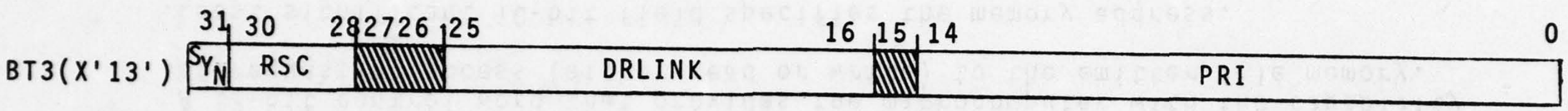
BP - Bypass PW Correlation

PW - Pulse Width

F - Frequency

TABLE B-V

BUS TRANSCEIVER LATCH 3



SYN - Resync Flag

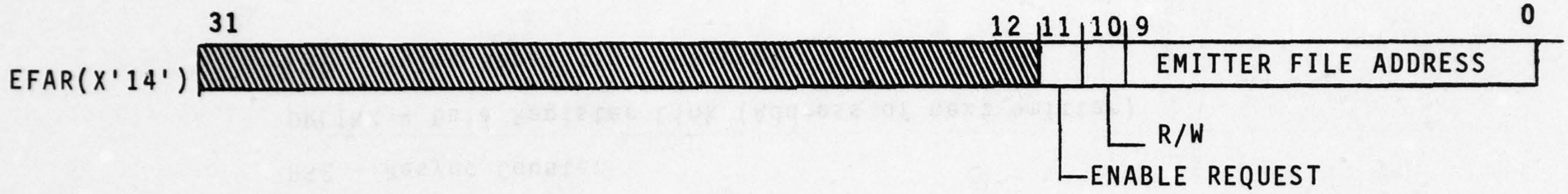
RSC - Resync Counter

DRLINK - Data Register Link (Address of next emitter)

PRI - Pulse Repetition Interval

TABLE B-VI

EMITTER FILE ADDRESS REGISTER



A 12-bit control word that provides the microcomputer with the capability of requesting access (either read or write) to the emitter file memory.

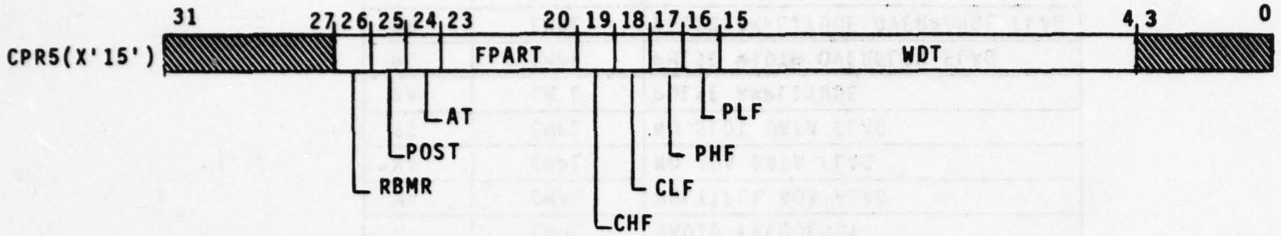
Least significant 10-bit field specifies the memory address.

R/W - 0 = Read Memory (i.e., Bus Transceiver Latch   Emitter File)
      1 = Write Memory (i.e., Emitter File   Bus Transceiver Latch)

Enable Request - Set command bit = 1 to enable memory read or write request. Command bit is cleared by the request logic upon completion of the memory access sequence.

TABLE B-VII

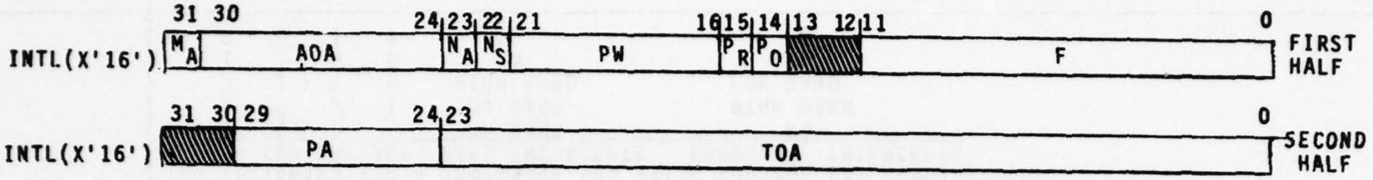
COMMAND PARAMETER & WDT LATCH



SYMBOL	FORM OF DATA (NORMAL OR COMPLEMENT)	COMMENTS
WDT	NORMAL	The 12-bit field contains the desired watchdog timer (WDT) interval. To load the field into the WDT countdown register requires that this write operation be preceded by a LOAD WDT command (see CPR7). When the register count reaches zero, an interrupt signal is sent to the microcomputer. Description of the WDT field: 1) 11-bit binary counter (bits 14-4) with bit 15 normally zero. 2) Maximum time interval = 131,008 sec (2047x64) where the counter resolution ≈ 64 sec. 3) Bit 15 = 1; disables the interval counter.
CHF, CLF, PHF, PLF	NORMAL NORMAL	These bits control a frequency comparator whereby each incoming pulse intercept is selected either for correlation against the emitter file words or passed directly to the intercept buffer (MAP global memory). Frequency parameter specified by the FPART field constitutes the criteria for partitioning into a high and low band. The following table indicates the valid control bit combinations: CHF CLF PHF PLF PASS INTERCEPTS CORRELATE INTERCEPTS 0 0 0 1 LOW BAND NO 1 0 0 1 LOW BAND HIGH BAND 0 1 1 0 HIGH BAND LOW BAND 1 1 0 0 NO ALL 0 0 1 1 ALL NO
FPART	NORMAL	Specifies the frequency for partitioning into low and high bands. FPART frequency field is compared against the four bits (8-5) of the pulse intercept frequency.
AT	NORMAL	M&T panel "AT" indicator. 0=Indicator off. 1=Indicator on.
POST	NORMAL	M&T panel "POST" indicator. 0=Indicator off. 1=Indicator on.
RBMR	NORMAL	Receiver Buffer Memory Reset 0=Clear Reset 1=Reset

TABLE B-VIII

INTERCEPT LATCH



Used to self test the preprocessor and MAP Interface. The preprocessor input register (IR) is normally driven by the receiver interface bus. Pulse intercept parameters are loaded into the two halfs of the IR thru two 32-bit data word transfers. The preprocessor may select the intercept latch as an alternative data source to the IR for self testing. This mode requires inhibiting the preprocessor, that is, issuing the command ENABLE PROCESSOR=0 (see CPR7). To load the IR requires two data transfers from the intercept latch. Each half intercept word is loaded in succession into the IR following the procedure of first, loading the intercept latch and secondly, issuing the command, LOAD BYTE=0 (see CPR7). The definition of the half intercept word formats are given above.

SYMBOL	FORM OF DATA (NORMAL OR COMPLEMENT)	
AOA	CMPL	ANGLE OF ARRIVAL
F	CMPL	RADIO FREQUENCY
MA	CMPL	MULTIPLE AOA FLAG
NA	CMPL	NO AOA DATA FLAG
NS	CMPL	NO SLOT DATA FLAG
PA	CMPL	PULSE AMPLITUDE
PO	CMPL	PULSE WIDTH OVERFLOW FLAG
PR	CMPL	PULSE AMPLITUDE OVERRANGE FLAG
PW	CMPL	PULSE WIDTH
TOA	CMPL	TIME OF ARRIVAL

TABLE B-IX

COMMAND PARAMETER LATCH

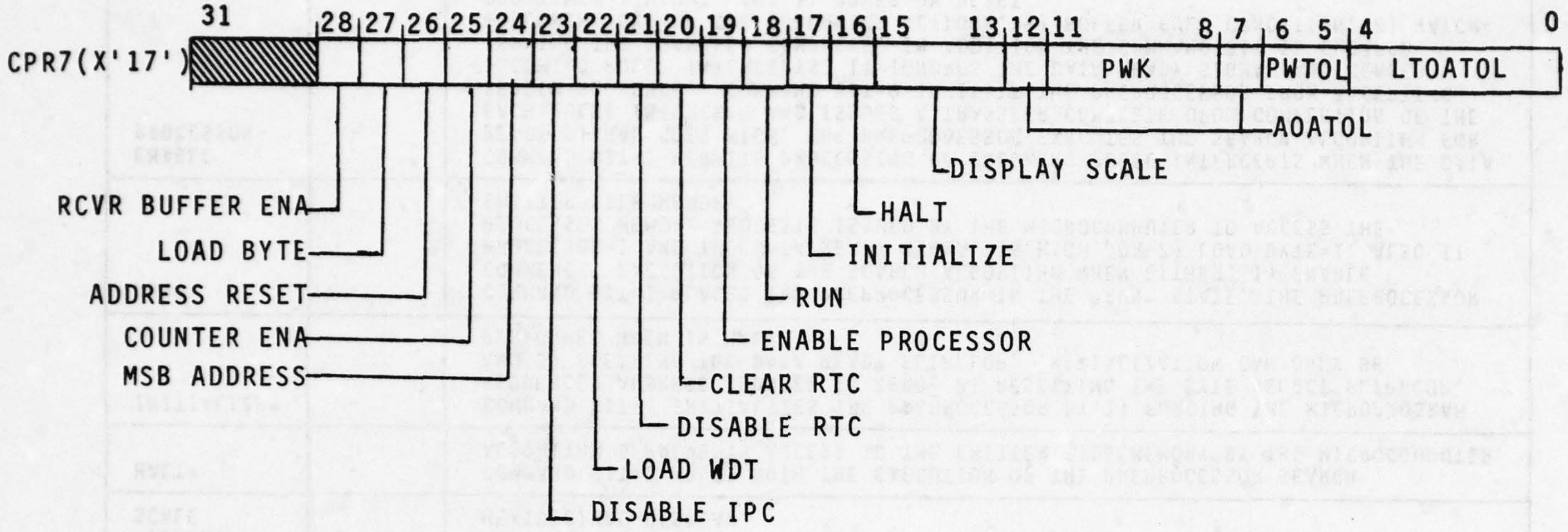


TABLE B-IX

COMMAND PARAMETER LATCH (cont'd)

LTOATOL	NORMAL	PROGRAMMABLE TOLERANCE LIMITS FOR TOA CORRELATION. MAXIMUM TOLERANCE COUNT=31.
PWTOL	NORMAL	PROGRAMMABLE TOLERANCE LIMITS FOR PW CORRELATION. MAXIMUM TOLERANCE COUNT=3.
AOATOL	NORMAL	PROGRAMMABLE TOLERANCE LIMITS FOR AOA CORRELATION. MAXIMUM TOLERANCE COUNT=3.
PWK	CMPL.	PULSE WIDTH BREAKPOINT
DISPLAY SCALE	CMPL.	CONTROLS THE SCALING (POSITION) OF THE DECIMAL POINT ON THE M&T PANEL HEXIDECIMAL DISPLAY.
HALT*	-	COMMAND BIT=1 HALTS BOTH THE EXECUTION OF THE PREPROCESSOR SEARCH ALGORITHM & PREVENTS ACCESS TO THE EMITTER FILE MEMORY BY THE MICROCOMPUTER.
INITIALIZE*	-	COMMAND BIT=1 INITIALIZES THE PREPROCESSOR BY 1) FORCING THE MICROPROGRAM SEQUENCER ADDRESS COUNTER TO ZERO, 2) RESETTING THE BYTE SELECT FLIPFLOP, AND 3) CLEARING THE DATA READY FLIPFLOP. INITIALIZATION CAN ONLY BE PERFORMED WHEN IN HALT.
RUN*	-	COMMAND BIT=1 PLACES THE PREPROCESSOR IN THE "RUN" STATE. THE PREPROCESSOR COMMENCES EXECUTION OF THE SEARCH ALGORITHM WHEN EITHER, 1) ENABLE PROCESSOR=1 AND THE DATA READY SIGNAL IS HIGH, OR 2) LOAD BYTE=1. ALSO IT PROCESSES MEMORY REQUESTS ISSUED BY THE MICROCOMPUTER TO ACCESS THE EMITTER FILE MEMORY.
ENABLE PROCESSOR	-	COMMAND BIT=1 PERMITS PROCESSING OF INCOMING PULSE INTERCEPTS WHEN THE DATA READY SIGNAL GOES HIGH. THE PREPROCESSOR EXECUTES THE SEARCH ALGORITHM FOR EACH PULSE INTERCEPT AND ISSUES A TRANSFER COMPLETE UPON COMPLETION OF THE SEARCH SEQUENCE. COMMAND BIT=0 INHIBITS THE PREPROCESSOR FROM ACCEPTING INCOMING PULSE INTERCEPTS. IT IGNORES THE DATA READY SIGNAL AND CEASES ISSUING THE TRANSFER COMPLETE. IN ADDITION THE COMMAND BIT IS CLEARED FOLLOWING ONE OF THE FOLLOWING ACTIONS, 1) BUFFER FULL CONDITION, 2) WATCH-DOG TIMER TIMEOUT, AND 3) POWER-ON RESET.
CLEAR RTC	-	COMMAND BIT=0 RESETS THE REAL TIME CLOCK (RTC) AND INPUT PULSE COUNTER (IPC) TO ZERO. COMMAND BIT=1 ENABLES RTC AND IPC. SEE NOTE 4.
DISABLE** RTC	-	COMMAND BIT=0 INHIBITS THE RTC LATCH TO PERMIT READING THE RTC. THE RTC CONTINUES TO UPDATE. COMMAND BIT=1 ALLOWS THE RTC TO UPDATE THE RTC LATCH.

TABLE B-IX

COMMAND PARAMETER LATCH (end)

LOAD WDT	-	LOADING THE WATCHDOG TIMER (WDT) WITH THE DESIRED TIME INTERVAL REQUIRES TWO STEPS. FIRST, THE COMMAND BIT=0 ENABLES LOADING THE WDT REGISTER. SECOND, THE LOADING OF THE TIME INTERVAL REQUIRES A "WRITE" TO THE COMM.-PAR. & WDT LATCH (12 BIT FIELD). UPON COMPLETION OF THE "WRITE", THE TIMER COMMENCES OPERATION. SEE NOTE 4.
DISABLE IPC**	-	COMMAND BIT=0 INHIBITS THE UPDATE OF THE INPUT PULSE COUNTER (IPC) LATCH TO PERMIT READING THE CURRENT INPUT PULSE COUNT. THE IPC CONTINUES TO UPDATE AND IS CLOCKED BY THE TRANSFER COMPLETE SIGNAL. COMMAND BIT=1 ALLOWS THE IPC TO UPDATE THE IPC LATCH. SEE
MSB ADDRESS	-	CONTROLS THE MOST SIGNIFICANT BIT (MSB) OF THE INTERCEPT BUFFER ADDRESS COUNTER. BIT=0, MSB=0 BIT=1, MSB=1
COUNTER ENABLE	-	CONTROLS THE INTERCEPT BUFFER ADDRESS COUNTER. BIT=0, MSB=0 BIT=1, MSB=1
ADDRESS RESET	-	CONTROLS THE INTERCEPT BUFFER ADDRESS COUNTER RESET FUNCTION. COMMAND BIT=1 RESETS THE COUNTER AND MUST BE FOLLOWED BY A COMMAND BIT=0 TO CLEAR THE RESET.
LOAD BYTE	-	THIS COMMAND PERMITS LOADING PULSE INTERCEPT PARAMETERS INTO THE PREPROCESSOR INPUT REGISTER (IR) FROM THE INTERCEPT LATCH. THIS FEATURE IS USEFUL IN SELF-TEST MODE IS ENABLED WHEN THE ENABLE PROCESSOR COMMAND BIT=0. TO LOAD THE UPPER AND LOWER HALVES OF THE IR REQUIRES TWO LOAD CYCLES, EACH CONSISTING OF WRITING A HALF INTERCEPT WORD INTO THE INTERCEPT LATCH FOLLOWED BY ISSUING A LOAD BYTE COMMAND BIT=1. EXECUTION OF THE SEARCH ALGORITHM COMMENCES UPON COMPLETION OF THE SECOND LOAD CYCLE. AFTER EACH LOAD CYCLE, THE COMMAND BIT IS CLEARED BY THE PREPROCESSOR FOLLOWING THE TRANSFER OF DATA TO THE IR.
RCVR BUFFER ENABLE	-	CONTROLS OPERATION OF THE RECEIVER BUFFER. BIT=0, INHIBITS BUFFER. BIT=1, ENABLES BUFFER.

* THE COMMAND BIT IS SET TO THE "1" STATE FOR A SINGLE ITERATION AND SUBSEQUENTLY CLEARED BY A HARDWARE RESET UPON COMPLETION OF THE COMMAND SEQUENCE.

** THE HARDWARE RESTORES THE COMMAND BIT TO THE NORMAL "1" STATE BY ANY OF THE FOLLOWING ACTIONS:
1) ISSUE A "READ" TO THE KEYBOARD REGISTER (OR IPCSR).
2) ISSUE A "READ" TO THE RTC REGISTER.
3) ISSUE A "WRITE" TO THE COMM.-PAR. & WDT LATCH.

TABLE B-X

KEYBOARD REGISTER

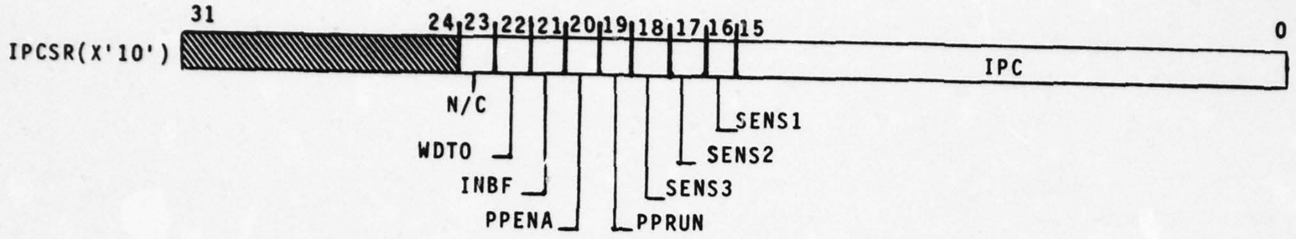
	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
KBR(X'10')	HEX7	HEX6	HEX5	HEX4	HEX3	HEX2	HEX1	HEX0
	MSB							LSB

The Keyboard register is loaded from the M&T panel Keyboard with a string of eight hexidecimal characters. The microcomputer may be programmed to access data in the keyboard register. (See Note 5.)

Since the keyboard register (KBR) and the input pulse counter-status register (IPCSR) whare a common device address, the register selection is determined by the DISABLE IPC command bit. Normally the KBR is the source to the data bus for the read operation. To select the IPCSR as the source, the CPU "read" is preceded by a clear command bit (i.e., DISABLE IPC=0).

TABLE B-XI

INPUT PULSE COUNTER & STATUS REGISTER

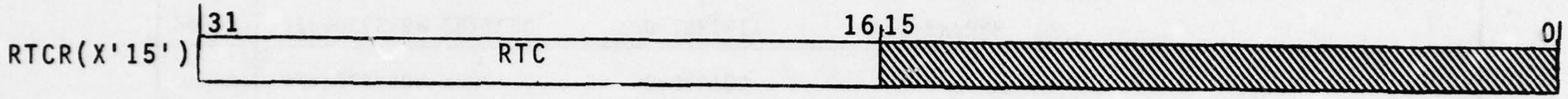


Name	Comments		
IPC	THE INPUT PULSE COUNTER (IPC) COUNTS INCOMING PULSE INTERCEPTS FROM THE RECEIVER SYSTEM. TO READ THE 16-BIT IPC REQUIRES THAT THE READ OPERATION BE PRECEDED BY A DISABLE IPC=0 COMMAND THAT TEMPORARILY SUSPENDS THE UPDATE OF THE IPC LATCH TO PERMIT READING THE IPC.		
SENS1*	M&T PANEL SENSE 1 SWITCH.	0=SWITCH DOWN	1=SWITCH UP
SENS2*	M&T PANEL SENSE 2 SWITCH.	0=SWITCH DOWN	1=SWITCH UP.
SENS3*	M&T PANEL SENSE 3 SWITCH.	0=SWITCH DOWN	1=SWITCH UP.
PPRUN*	PREPROCESSOR RUN	0=HALTED	1=RUN.
PPENA*	PREPROCESSOR ENABLED	0=INHIBIT	1=ENABLE
INBF*	INTERCEPT BUFFER FULL	0=BUFFER NOT FULL	1=BUFFER FULL (i.e., INTERCEPT BUFFER ADDRESS COUNTER=X'3FF').
WDTO*	WATCHDOG TIMEOUT	0=COUNTER≠0	1=COUNTER=0

* THIS DEVICE ADDRESS CODE (X'10') IS SHARED BY THE KBR AND IPCSR. NORMALLY A "READ" OPERATION ACCESSES THE KBR. TO SELECT THE IPCSR FOR A "READ" REQUIRES THAT IT BE PRECEDED BY A DISABLE IPC=0 COMMAND.

TABLE B-XII

REAL TIME CLOCK REGISTER

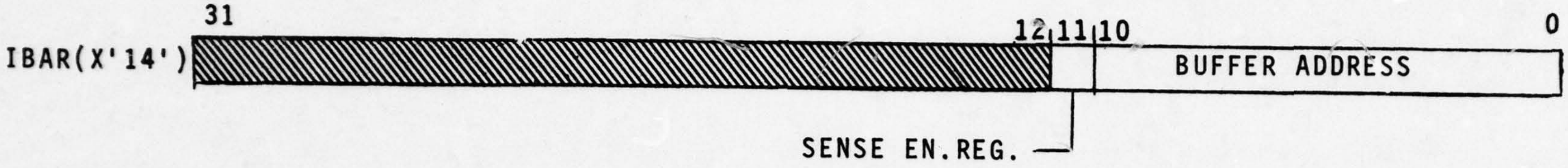


A 16-bit real time clock (RTC) latch contains the binary representation of the RTC.
Maximum value - 33,554,432 usec.
Resolution - 512 usec.

The read operation is preceded by a DISABLE RTC=0 command that temporarily suspends the update of the RTC latch to permit reading the RTC.

TABLE B-XIII

INTERCEPT BUFFER ADDRESS



An 11-bit field providing the intercept buffer address of the last pulse intercept passed to the intercept buffer (MAP global memory) by the preprocessor.

Sense En.Reg. - Indicates the status of the Memory Request Enable command.

0=Request Cleared 1=Request Active

☆U.S.Government Printing Office: 1979 — 657-002/612