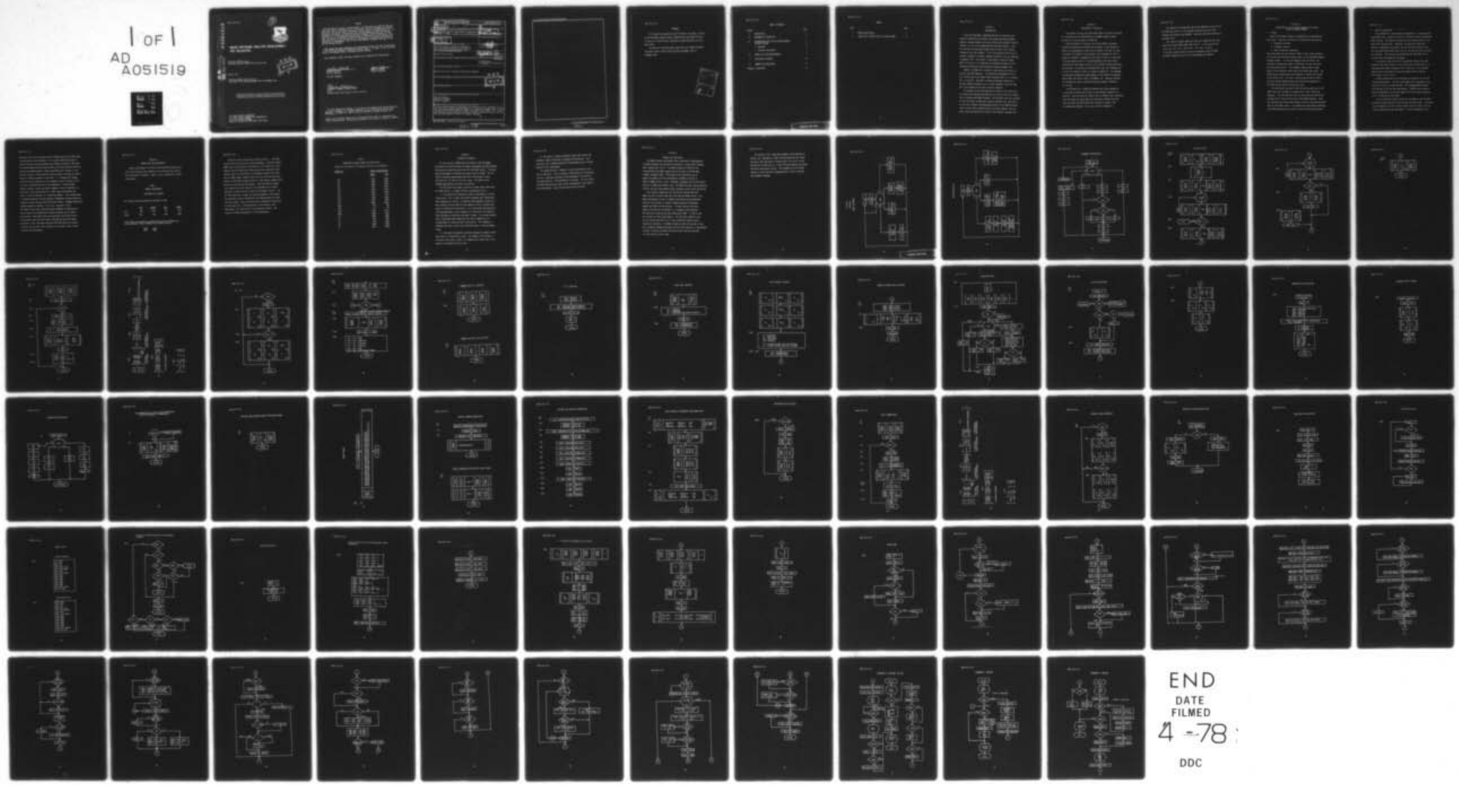
AD-A051 519 AIR FORCE AVIONICS LAB WRIGHT-PATTERSON AFB OHIO F/G 17/7
GEANS SOFTWARE ANALYSIS DEVELOPMENT AND VALIDATION.(U)
AUG 77 W MIKULSKI, J E BARNES
UNCLASSIFIED AFAL-TR-77-119 NL

1 OF 1
AD
A051519



END
DATE
FILMED
4-78
DDC

AFAL-TR-77-119

2

GEANS SOFTWARE ANALYSIS DEVELOPMENT AND VALIDATION

Reference Systems Branch
Reconnaissance and Weapon Delivery Division

August 1977

TECHNICAL REPORT AFAL-TR-77-119
FINAL REPORT FOR PERIOD NOVEMBER 1975 TO NOVEMBER 1976

Approved for public release; distribution unlimited.

AIR FORCE AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

William Mikulski
WILLIAM MIKULSKI
Project Engineer

James E Barnes
JAMES E. BARNES
Project Engineer

FOR THE COMMANDER

Charles L Hudson
CHARLES L. HUDSON, Colonel, USAF
Chief
Reconnaissance and Weapon Delivery Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFAL/RWA ,W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

14

1. REPORT NUMBER
AFAL-TR-77-119

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

9

4. TITLE (and Subtitle)

6

GEANS SOFTWARE ANALYSIS DEVELOPMENT AND VALIDATION.

5. TYPE OF REPORT & PERIOD COVERED
Final Rep[redacted]t
Nov [redacted]75 - Nov [redacted]76

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

10

William Mikulski
James E. Barnes

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Air Force Avionics Laboratory (AFAL/RWA-3)
Air Force Wright Aeronautical Laboratories
AFSC, Wright-Patterson AFB, Ohio 45433

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

16 17

Project 19270203

11. CONTROLLING OFFICE NAME AND ADDRESS
Same as block 9 above

12. REPORT DATE

11

Aug[redacted]77

12

13. NUMBER OF PAGES
~~79~~ 79 p.

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

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

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)

Computer program
High order language
Assembly language

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

The GEANS Software Analysis Development and Validation effort consisted of writing an existing assembly language program in a high order language (HOL) and then comparing the two-load modules for memory requirements and timing. This comparison showed that the program as written in the HOL required more memory and time than the same program as written in assembly language.

011 670

blank

AFAL-TR-77-119

FOREWORD

This report was prepared by William Mikulski and James E. Barnes of the Reference Systems Branch, Reconnaissance and Weapon Delivery Division, Air Force Avionics Laboratory, Wright-Patterson Air Force Base, Ohio.

The work was initiated under Project Work Unit Number 19270203. The report covers effort during the period November 1975 to November 1976.

TABLE OF CONTENTS

TABLES

SECTION I

INTRODUCTION

In May 1973 AFAL/RWA-3 undertook the task of converting the Honeywell Gimballed Electrostatic Gyro Navigation System (GEANS) computer Program program, which was hosted on the Honeywell HDC-601 computer, to run on the Singer/Kearfott SKC-2000 computer. Both the original and converted programs were written in the assembly language of the host computer. Thus, the conversion effort was an assembly language to assembly language conversion, and was completed in December 1975. This effort is described in Technical Report AFAL-TR 77-8, Vol 1 & 2, Conversion of Computer Software for the Gimballed Electrostatic Gyro Navigation System, November 1976.

GEANS is intended for a variety of applications using any of several available computers. The required reprogramming will be most rapidly done using Higher Order Languages (HOL), if a suitable HOL is available. Good HOL's increase programmer productivity, provide readable programs, and, for some languages, allow the same HOL source program to be used on several computers.

The purpose of the investigation described in this report was to evaluate the problems and penalties of using an avionics-oriented HOL to reprogram the GEANS software. The available time and resources limited the effort to applying a readily available HOL, J3B level 0, to only the Alignment and Navigation protions of the GEANS Software. These portions were converted and the resulting computer time and memory required compared with those of the assembly language code.

SECTION II

BACKGROUND OF JOVIAL/J3B

The compiler used was the JOVIAL-3B/0 compiler written by Softech Inc. and hosted on the IBM-370/155 at the WPAFB computer center. Following is a short history of JOVIAL-3B:

When the Request for Proposal for the B-1 bomber system was prepared at Aeronautical Systems Division (ASD), it was decided to require the programming of the mission software to be done in a HOL. In October 1971 JOVIAL-73 was selected as the language for the B-1. However the JOVIAL-73 language was not ready in time so in June 1972 a reduced capability HOL of low implementation risks which would satisfy the minimum needs of the B-1 until JOVIAL-73 became available was designed. To achieve low implementation risk, this language was a version of JOVIAL-5, which is a subset of JOVIAL-73. The language was designed by Boeing and three members of the JOVIAL-73 committee, namely RADC, ASD (the B-1 SPO), and ABACUS, Inc. Whenever possible the language was made to conform with what was known of JOVIAL-73 at the time.

The contract for a JOVIAL-3B compiler was finally awarded to a contractor (Softech) who did not use the JOVIAL-5 compiler as a baseline. Thus the need for staying close to JOVIAL-5 was eliminated. Since much more was known about the direction of JOVIAL-73 by the time the JOVIAL-3B specification was finished in October 1972 (by Boeing and Softech), it was very similar to JOVIAL-73.

Two compilers for JOVIAL-3B, one for the IBM-360 and one for the SKC-2000 were written in the Automated Engineering Design (AED) language and hosted on the IBM-360. They were delivered to Boeing in September 1973.

The original reduced capability version of JOVIAL-3B has since come to be termed JOVIAL-3B/0 (J3B level 0). It has evolved into JOVIAL-3B/1 and, later, JOVIAL-3B/2. The level 2 compiler is the most current version, and can be targeted to the LC-4516D and M362F computers as well as to the SKC-2000 and IBM-370.

SECTION III

INTEGRATION OF THE HOL GEANS PROGRAM WITH EXISTING ASSEMBLY LANGUAGE PROGRAM

1. APPROACH

Coding of the GEANS software in JOVIAL/J3B was separated into three parts. These were functionally different and consisted of:

1) Navigation routines.

2) Alignment routines.

3) Matrix operation subroutines.

Since these were the only portions coded in J3B, it was necessary to link with the existing software written in the SKC-2000 assembly language (FOCAP). To minimize debugging time and effort, the navigation routines were coded in J3B almost line for line from the hand-coded routines. The alignment routines were written based on the flowcharts rather than on the FOCAP code itself. The FOCAP code was checked with the flowcharts to verify the accuracy of the details only. The matrix routines were written in J3B to minimize linkage problems which would have been encountered if the FOCAP written matrix routines were used.

The same labels were used in the J3B code as were used in the FOCAP code, which allowed for greater ease in both timing and debugging. The same data base was also used since the J3B written programs had to link with the hand-written FOCAP programs. Using the same data base caused some problems since the J3B generated code does not use FOCAP "common." This problem was alleviated by modifying the FOCAP-Coded data base to refer to the J3B defined data blocks.

2. PROBLEMS ENCOUNTERED

Since JOVIAL/J3B is target machine independent, it cannot generate input/output code. Because of this, all I/O must be done by handwritten FOCAP routines. The required routines for I/O already existed in the GEANS real-time executive. Therefore, to provent the additional problem of debugging I/O routines, the J3B code was written to look like the hand-written FOCAP code as far as the executive (hand-written) was concerned. This was done by maintaining the same names for all routines which the executive referenced.

The executive code also had to be modified slightly since J3B generates code using "Page 3", which allows short instructions to be generated which access constants 0 and 1 from core. This required adding code in the executive to set the status word to indicate that a page 3 was in effect.

Another modification to the executive was required for the J3B linkage mechanization. J3B uses a push-down stack arrangement using registers 6 and 14 as stack pointers. The existing GEANS software used register 6 only for the same purpose. Therefore the executive had to be modified to initialize register 14 to the same as register 6 (i.e. to the top of the stack).

The existing math library written for the GEANS in the SKC-2000 had been previously proven to work properly and efficiently. Therefore, it was decided to use this library of math functions rather than the library supplied with the J3B compiler. The only drawback to this method was that the job linkage and the linkage used by the FOCAP coded library were entirely different. This required additional code to convert from one calling convention and linkage structure to the other. The hand-written math library was written to be most efficient with the hand-written GEANS program. Rather than modify this library, it was decided to keep it intact and add a buffer stage of routines to convert from one linkage convention to the other. The standard J3B linkage convention passes addresses of all arguments in Reg A and Reg B, with all others following the call to the subroutine. The hand-coded library, however, passed the actual argument in the A-B registers in some cases, used index register 4 in other cases and returned the result in the A-B register, or a specified address. This inconsistency in linkage procedures required separate intermediate linkage conversion routines for each routine used in the math library. Standard JOVIAL/J3B linkage convention requires saving index registers 1 thru 5 (XR1-XR5) on the stack, obtaining proper arguments or addresses and placing them at the appropriate location according to the routine being called. Upon return from the subroutine, the results had to be returned to the J3B caller where proscribed by the J3B linkage convention. Also, the index registers (XR1-XR5) had to be restored from the stack and the stack cleaned up to the state it was in when called by the J3B program.

SECTION IV
MEMORY AND TIME REQUIREMENTS

Memory requirements for both the J3B and FOCAP programs were taken from the memory map produced by the Linkage Editor step of the compilation or assembly. Table 1 gives a comparison of these requirements.

TABLE 1

MEMORY REQUIREMENTS

(SKC 2000 32 Bit Words)

For routines converted directly from FOCAP to J3B:

	NAV	ALIGN	INIT	TOTAL
J3B	885	1477	744	3106
FOCAP	1014	836	694	2544

Total memory requirements, including the executive (RTEXEC) and math subroutine library (SUBLIB) plus linkage routines:

FOCAP	4524
J3B	6350

Timing for these programs are listed in Table 2. All major routines were timed using the following method: A Hewlett-Packard 1600S Logic State Analyzer, consisting of a H-P 1600A Logic State Analyzer and a HP 1607 Logic State Analyzer was connected to the address lines of the SKC-2000 at the Computer Control Unit (CCU). The H-P 1600A address compare lines were set to the desired start address of the GEANS program and the H-P 1607 address compare lines were set to the desired stop address. When the address compare lines matched the start address in the SKC-2000 the H-P 1600A generated a trigger which was fed to an electronic counter. The H-P 1607 generated a trigger for the stop address in like manner. The electronic counter recorded the time delay between the start and stop triggers, thus giving the time of execution of the block of code under test. The uncertainty in this method is 100 nanoseconds, the sensitivity of the electronic counter. The accuracy of these measurements is ±100 nanoseconds.

TABLE 2

COMPARISON OF GEANS TIMING IN J3B AND FOCAP

(Subroutine description in flowchart form are in the Appendix.)

SUBROUTINE	TIME IN MICROSECONDS	
	FOCAP	J3B
IIC	1337	3000
IID	2620	4500
IIE	1250	2750
IIF	8417	16975
IIG	3480	12204
IIH	3162	4670
IIK	1183	1756
IIM	1636	2542
IIO	3642	6318
IIP	13066	47345
RSET	347	730
IIR	45	39
FENT	6676	17151
DECD	2340	2440
IE	2460	3900
IF	977	1685
IG	790	1685
IH	3883	5400
IJ	7254	7800

SECTION V

DISCUSSION OF RESULTS

For that portion of GEANS that was written in J3B (Alignment, Navigation and Initialization) the memory requirement was approximately 20% greater for the J3B version than for the FOCAP version. The total memory requirement for J3B was 29% greater than for FOCAP. The J3B compiler produced approximately 80% short instructions (a short instruction is 1/2 word, or 16 bits, long). The FOCAP version produced approximately 15% short instructions.

The J3B version was found to run two to three times slower than the FOCAP version. A number of things contributed to this:

1) The compiler was designed to create a high density of short instructions. This it does, and actually produces more instructions than necessary as a result. It reserves Index Register 5 (XR5) as a base register and loads XR5 with the address of one of several data areas to create short instructions. As a result when some operations are being performed, such as creation of a 3x3 matrix, the compiler loads XR5 once for each data item that is moved. So an index register load is performed (which takes one full word of memory and 2.5 microseconds) to create two short instructions. This sequence is repeated nine times, and is very inefficient both in time and memory usage.

2) The compiler generates code which computes an address offset each time it is used within a loop. For example, if the offset is used four times within a loop it is computed four times, and it only needs to be computed the first time.

3) The special linkage subroutines, those that resolve the J3B/FOCAP linkage differences, consumed 50 microseconds. Each subroutine call in GEANS required 100 microseconds more to execute because of this requirement.

The JOVIAL-3B level 0 compiler is poorly optimized to save time of execution. It will produce a high density of instructions, and with a data base designed to take advantage of this ability some optimization in time and memory is possible. For this effort the FOCAP data base was used, so the shortcomings of the compiler were accentuated. Later versions may be more efficient.

SECTION VI

SUMMARY AND CONCLUSIONS

The GEANS software development effort consisted of reprogramming the GEANS Alignment and Navigation algorithms in a High Order Language (HOL), JOVIAL/J3B, Level 0. The GEANS program was hosted on the Singer/Kearfott SKC-2000 computer and was written in the SKC-2000 assembly language, FOCAP. The purpose of the J3B effort was to compare the memory and time requirements of GEANS as written in FOCAP to GEANS written in J3B. Alignment, navigation and initialization portions of GEANS were coded in J3B. The FOCAP data base, math subroutine library, and real time executive were retained as part of the J3B version.

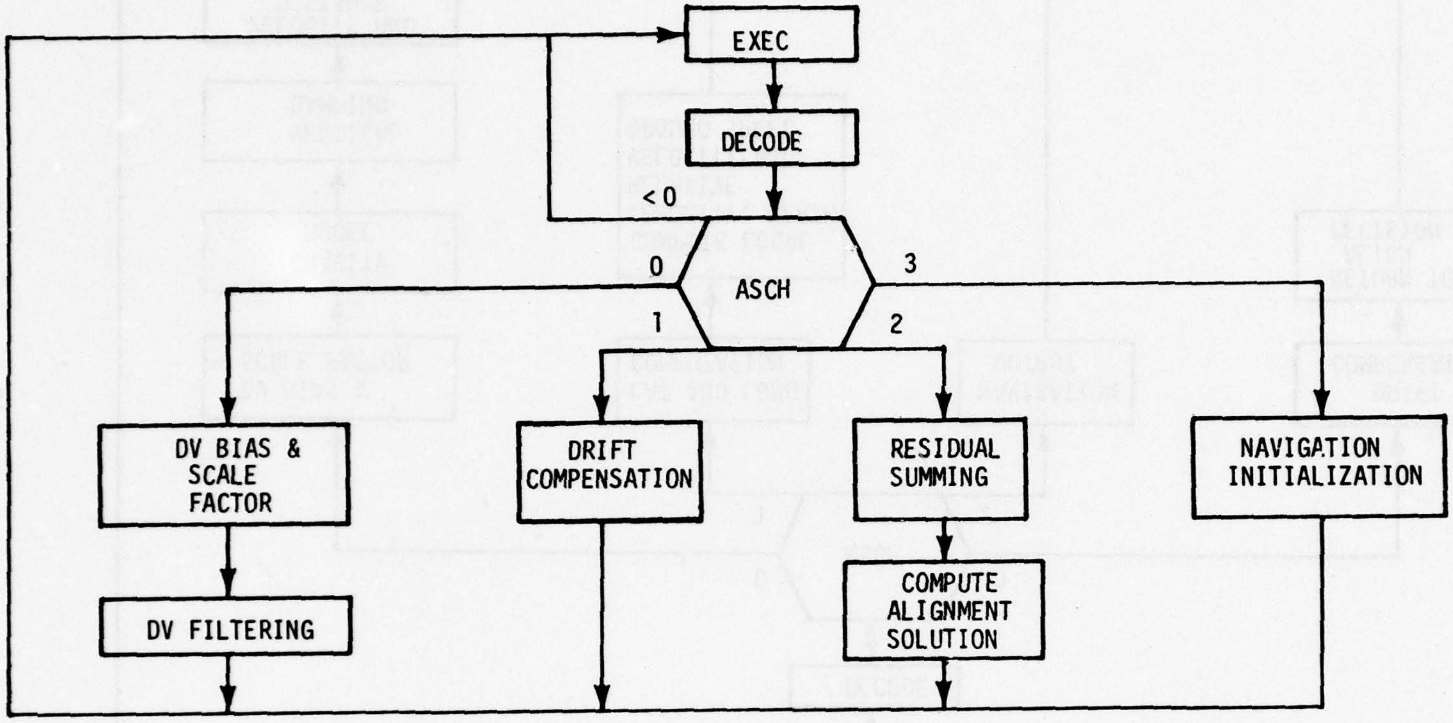
Final results showed that the J3B version required 29% more memory and two to three times more time than the FOCAP version. The memory requirement of 29% is somewhat misleading because additional code had to be written to resolve linkage convention differences between the FOCAP and J3B versions. In those routines that were coded in J3B directly from FOCAP (i.e. Alignment and Navigation) the J3B version required 20% more memory than FOCAP. A similar case can be made for timing requirements. The particular compiler used for this study (J3B Level 0) is a very inefficient compiler and is very poorly optimized. If GEANS had been written from scratch in J3B, with a properly designed data base and real time executive, a considerable increase in timing and memory efficiency might have been realized ~ver the results of this study.

The results of this study were brought to the attention of Softech, Inc. (designers of J3B), who explained that the latest version of J3B (J3B Level 2) corrects most, if not all, of the deficiencies of J3B level 0. A more efficient compiler would most definitely show better results. The overhead for an efficient compiler of any high order language would be close to 20% more than assembly language.

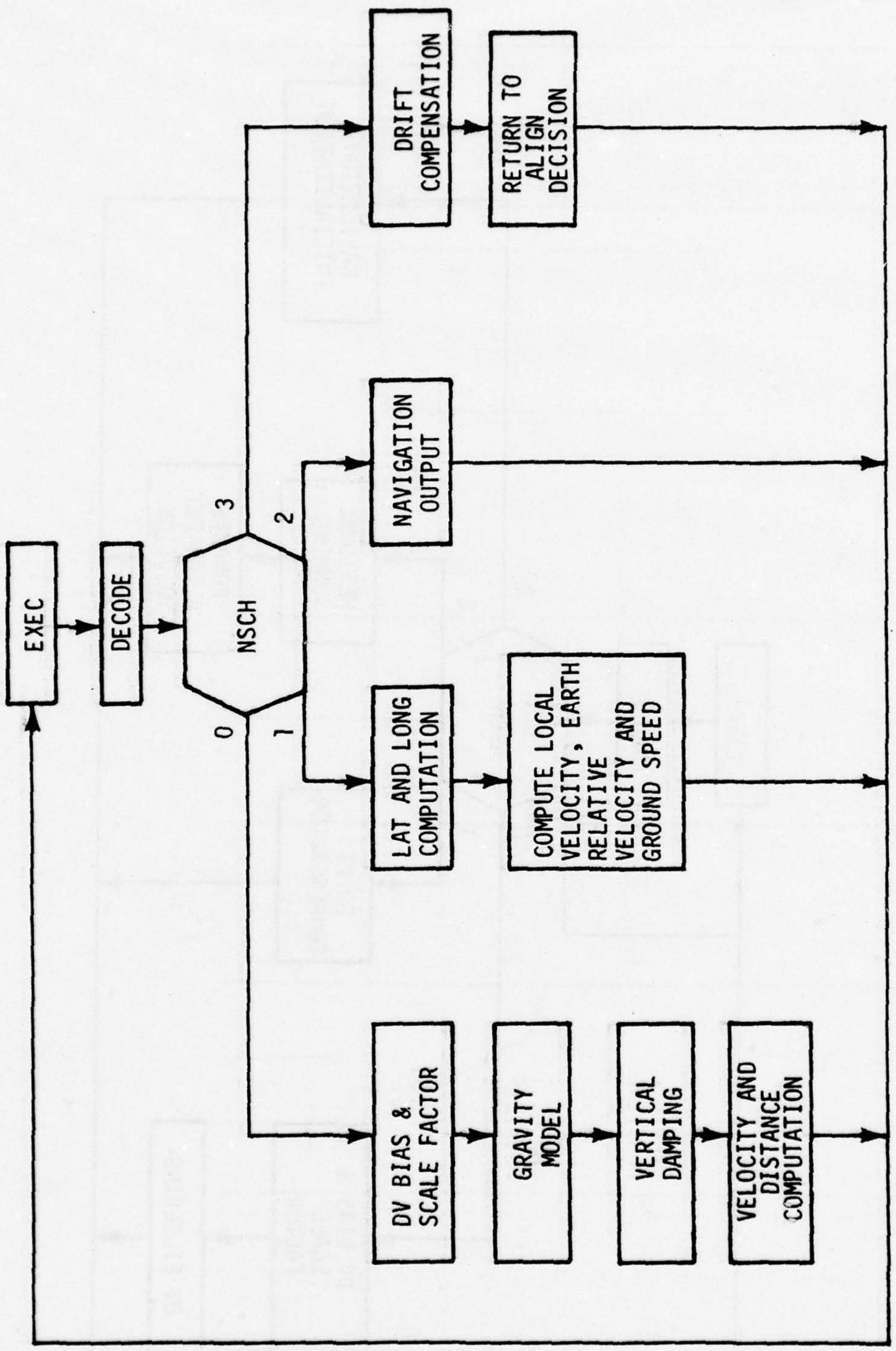
APPENDIX

FLOWCHARTS

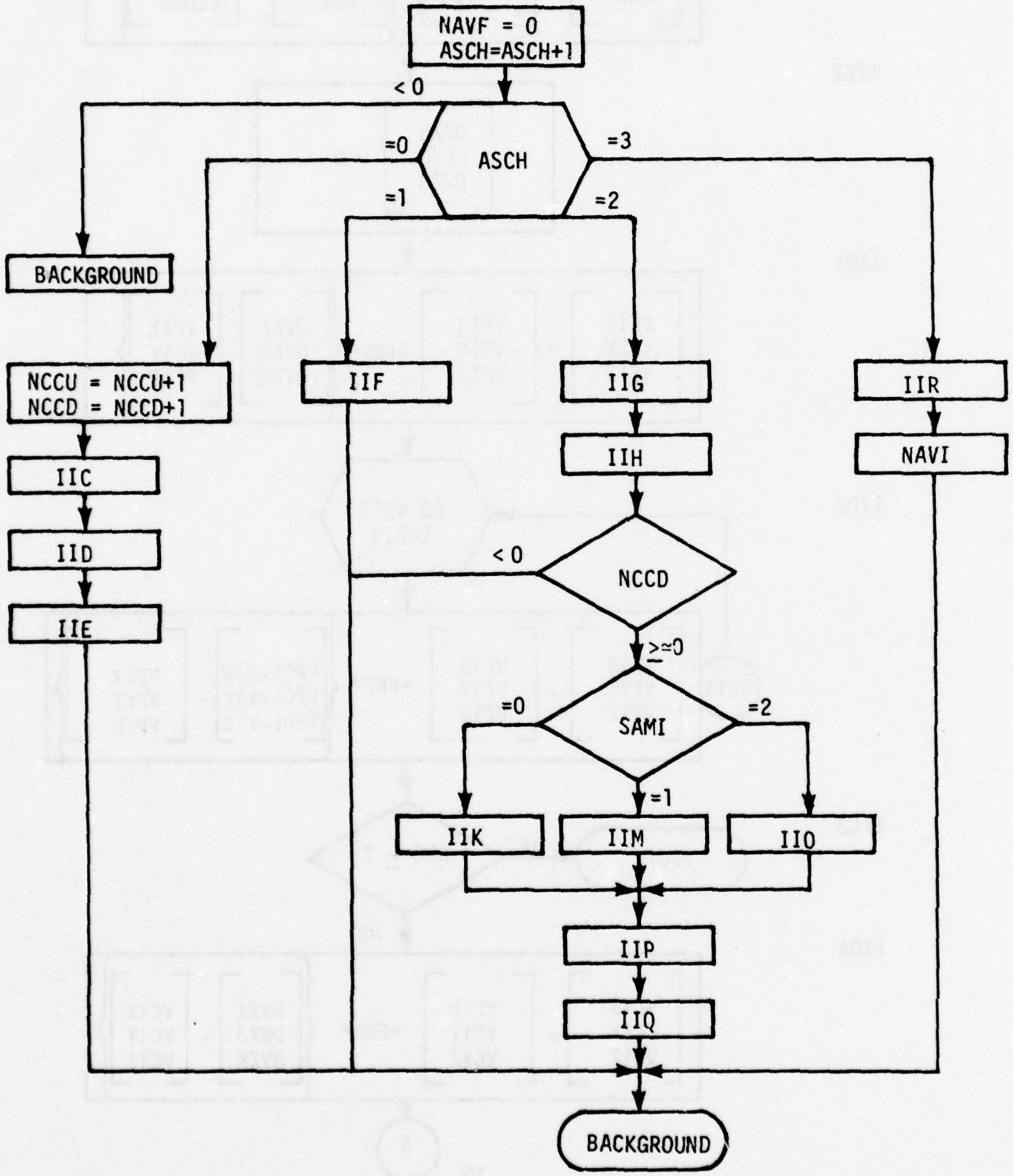
GEANS ALIGNMENT



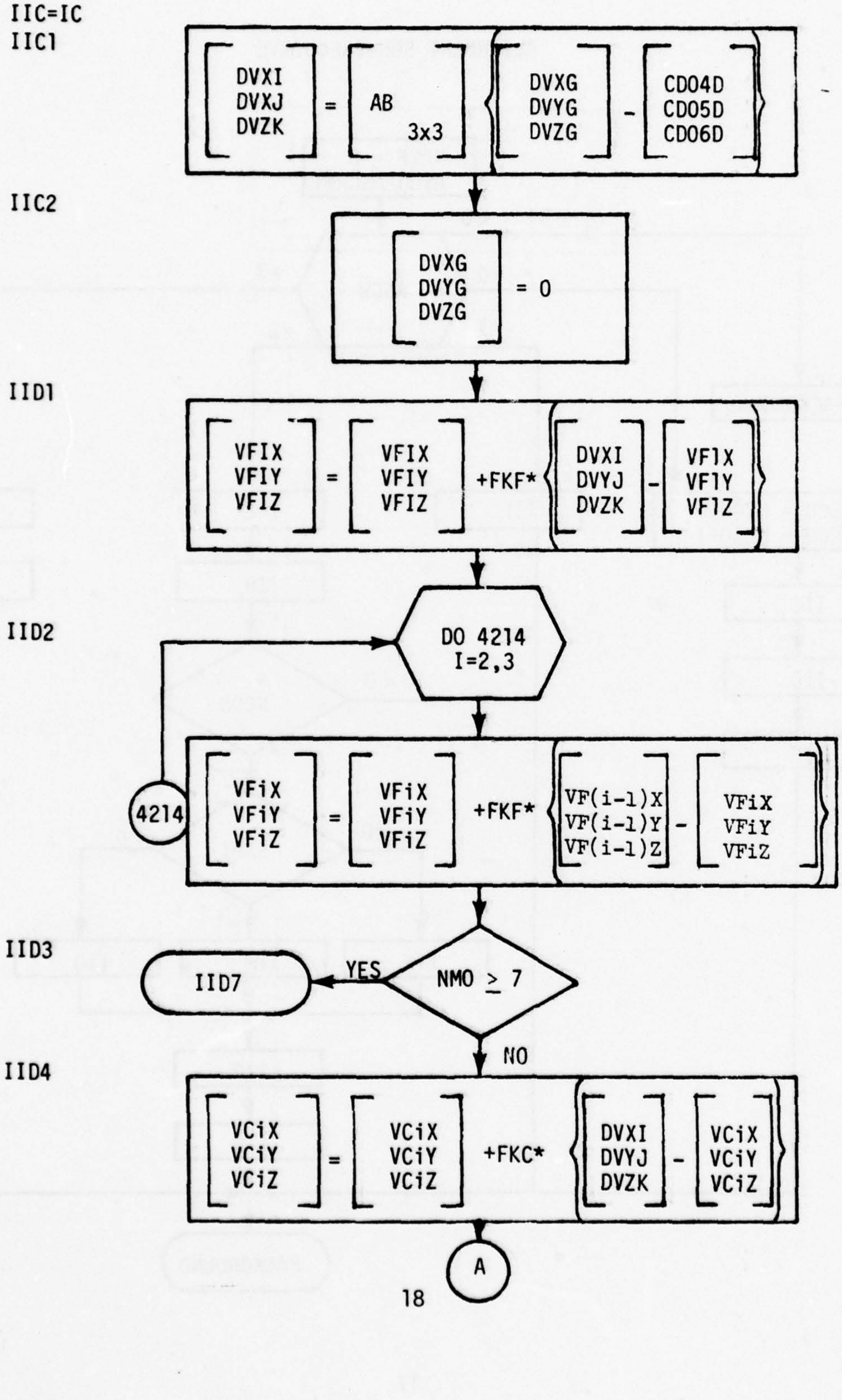
GEANS NAVIGATION

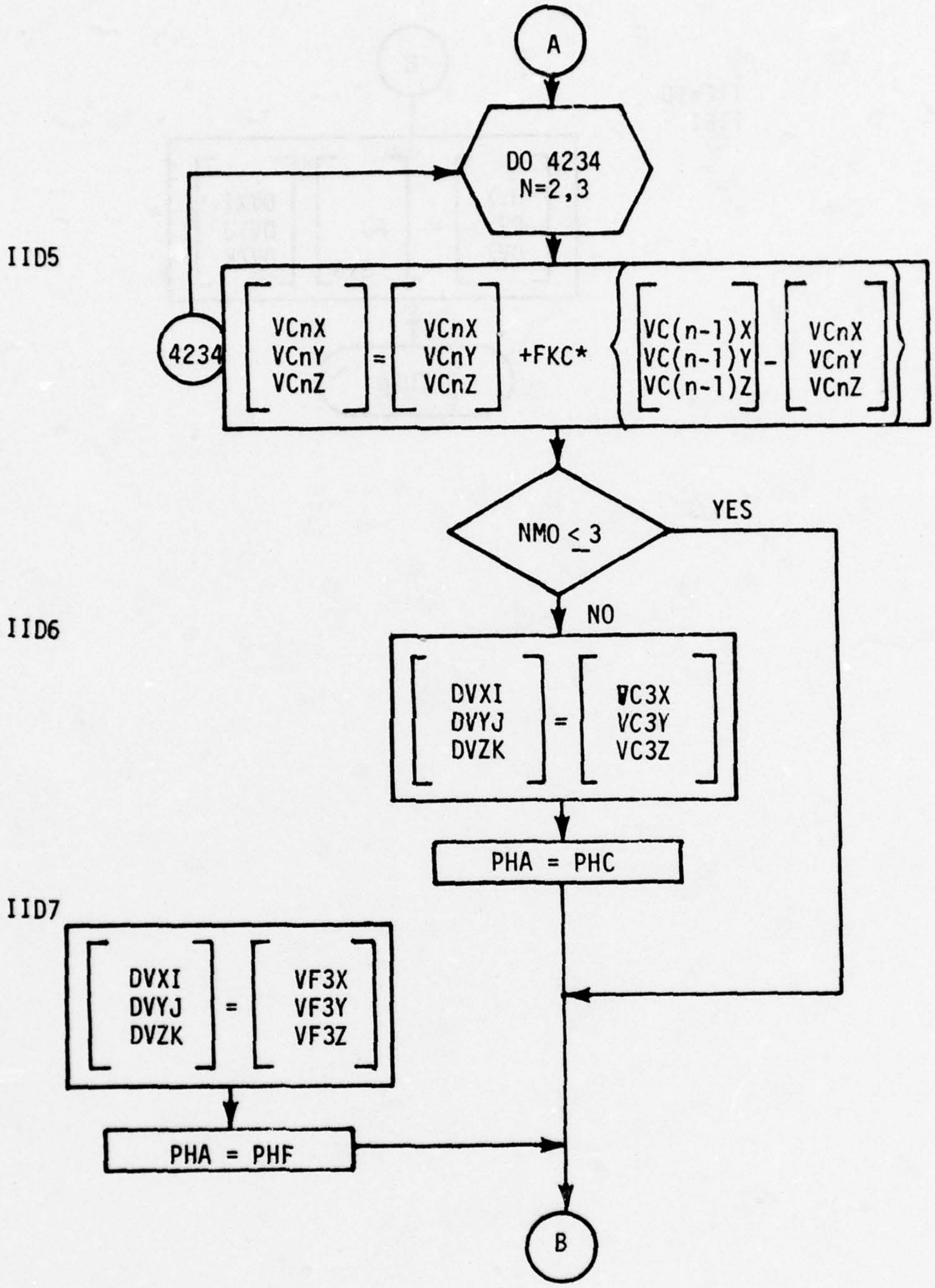


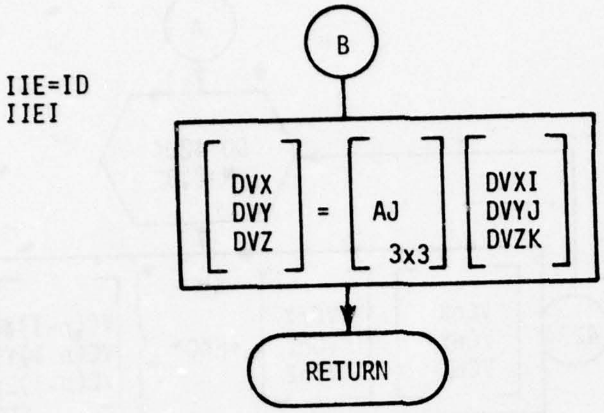
ALIGNMENT SUB-EXECUTIVE

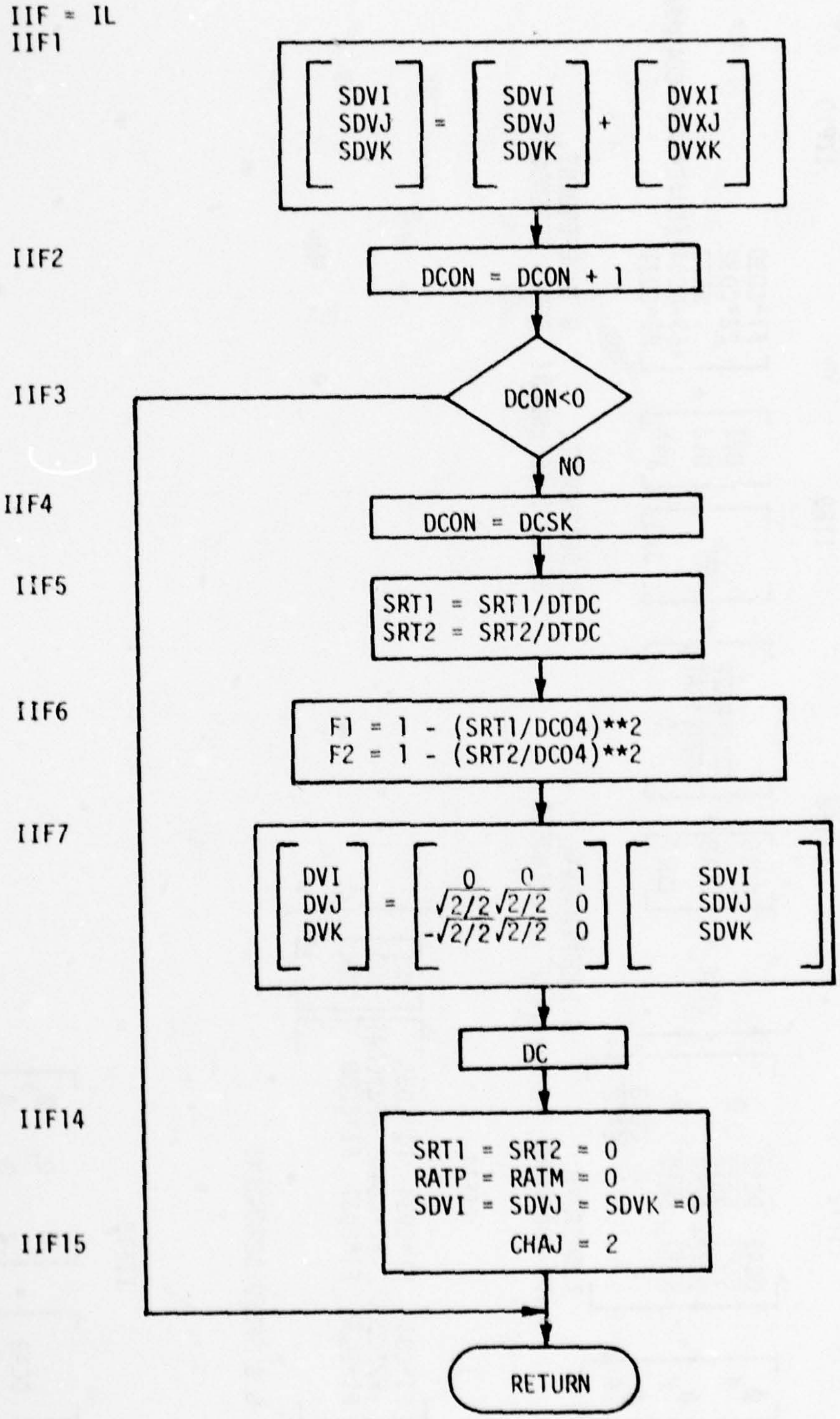


LOW PASS FILTER


IIC=IC
IIC1
DVXI DVXJ DVZK = AB 3x3 {DVXG DVYG DVZG - CD04D CD05D CD06D}
IIC2
DVXG DVYG DVZG = 0
IID1
VFIX VFIY VFIZ = VFIX VFIY VFIZ +FKF* {DVXI DVYJ DVZK - VF1X VF1Y VF1Z}
IID2
DO 4214 I=2,3
4214
VFiX VFiY VFiZ = VFiX VFiY VFiZ +FKF* {VF(i-1)X VF(i-1)Y VF(i-1)Z - VFiX VFiY VFiZ}
IID3
IID7
YES
NMO ≥ 7
NO
IID4
VCiX VCiY VCiZ = VCiX VCiY VCiZ +FKC* {DVXI DVYJ DVZK - VCiX VCiY VCiZ}
A


A
DO 4234
N=2,3
IID5
4234
VCnX
VCnY
VCnZ
=
VCnX
VCnY
VCnZ
+FKC*
VC(n-1)X
VC(n-1)Y
VC(n-1)Z
-
VCnX
VCnY
VCnZ
NMO ≤ 3
YES
NO
IID6
DVXI
DVYJ
DVZK
=
VC3X
VC3Y
VC3Z
PHA = PHC
IID7
DVXI
DVYJ
DVZK
=
VF3X
VF3Y
VF3Z
PHA = PHF
B


B
IIE=ID
IIEI
DVX
DVY
DVZ
=
AJ
3x3
DVXI
DVYJ
DVZK
RETURN


IIF = IL
IIF1
SDVI SDVJ SDVK = SDVI SDVJ SDVK + DVXI DVXJ DVXK
IIF2
DCON = DCON + 1
IIF3
DCON<0
NO
IIF4
DCON = DCSK
IIF5
SRT1 = SRT1/DTDC
SRT2 = SRT2/DTDC
IIF6
F1 = 1 - (SRT1/DCO4)**2
F2 = 1 - (SRT2/DCO4)**2
IIF7
DVI DVJ DVK = 0 0 1 ; √2/2 √2/2 0 ; -√2/2 √2/2 0 ; SDVI SDVJ SDVK
DC
IIF14
SRT1 = SRT2 = 0
RATP = RATM = 0
SDVI = SDVJ = SDVK =0
IIF15
CHAJ = 2
RETURN

IIF12 IIF8 IIF9 IIF10

$$\begin{bmatrix} Ø_x \\ Ø_y \\ Ø_z \end{bmatrix} = \begin{bmatrix} \frac{DC42}{SRT1} & \frac{DC42}{SRT2} & 0 \\ \frac{DC42}{SRT1}- & \frac{DC42}{SRT2} & 0 \\ 0 & 0 & \frac{DC43}{SRT2} \end{bmatrix} \left\{ DTDC * \begin{bmatrix} CD16 \\ CD18 \\ CD17 \end{bmatrix} + \begin{bmatrix} 0 \\ CD28*RATP \\ -CD29*RATM \\ 0 \end{bmatrix} + \begin{bmatrix} GM \\ \quad 3x3 \end{bmatrix} \begin{bmatrix} DVI \\ DVJ \\ DVK \end{bmatrix} + \begin{bmatrix} F1*CD30 \\ F2*CD32 \\ SRT2 \\ +(1-DC04)(CD42*RATP-CD43*RATM) \\ F1*CD31 \end{bmatrix} \right.$$

TRANSFORM TO PLATFORM

G INDEPENDENT, SPEED INDEPENDENT

G DEPENDENT, SPEED INDEPENDENT

G INDEPENDENT, SPEED DEPENDENT

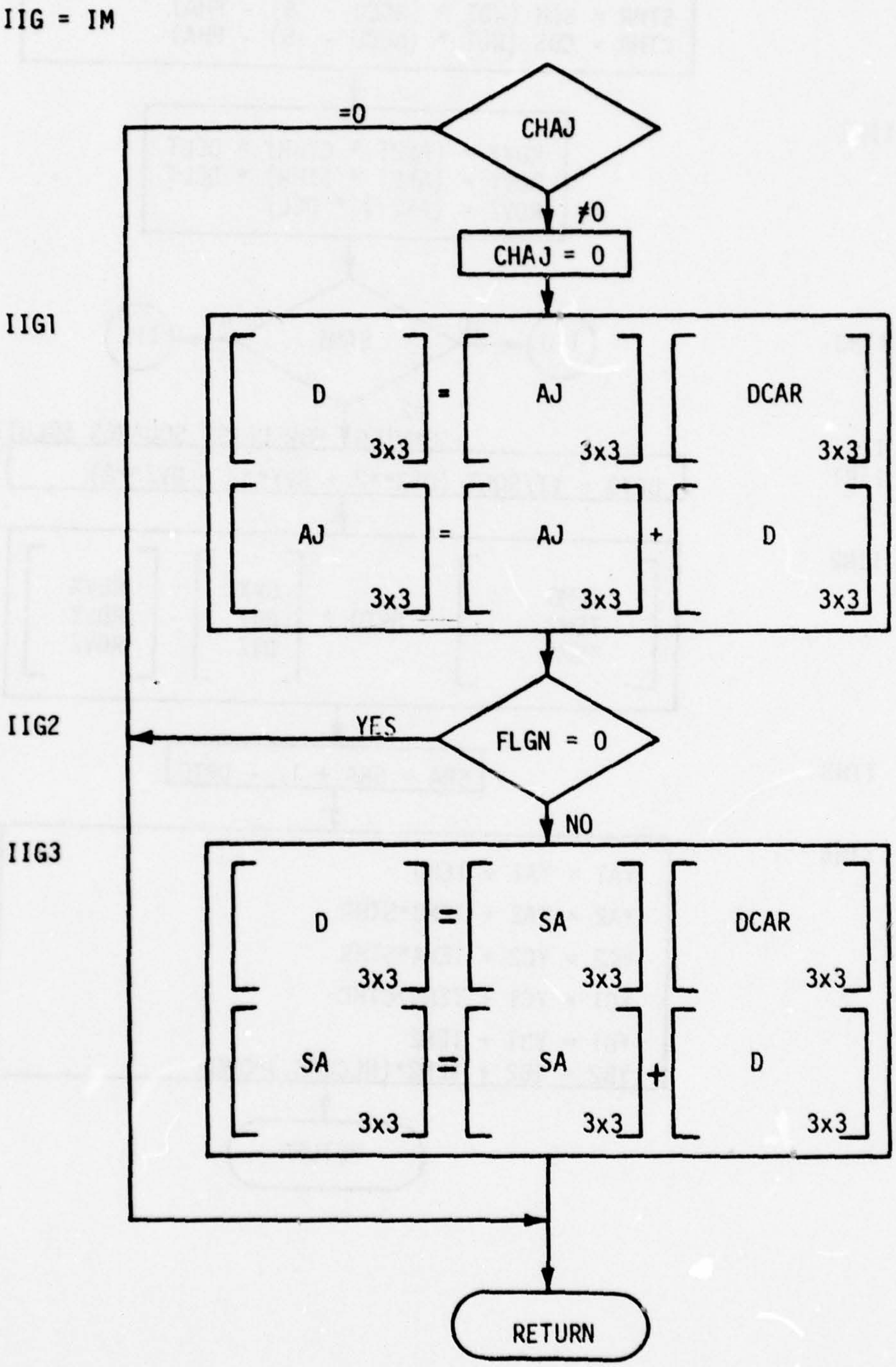
IIF11

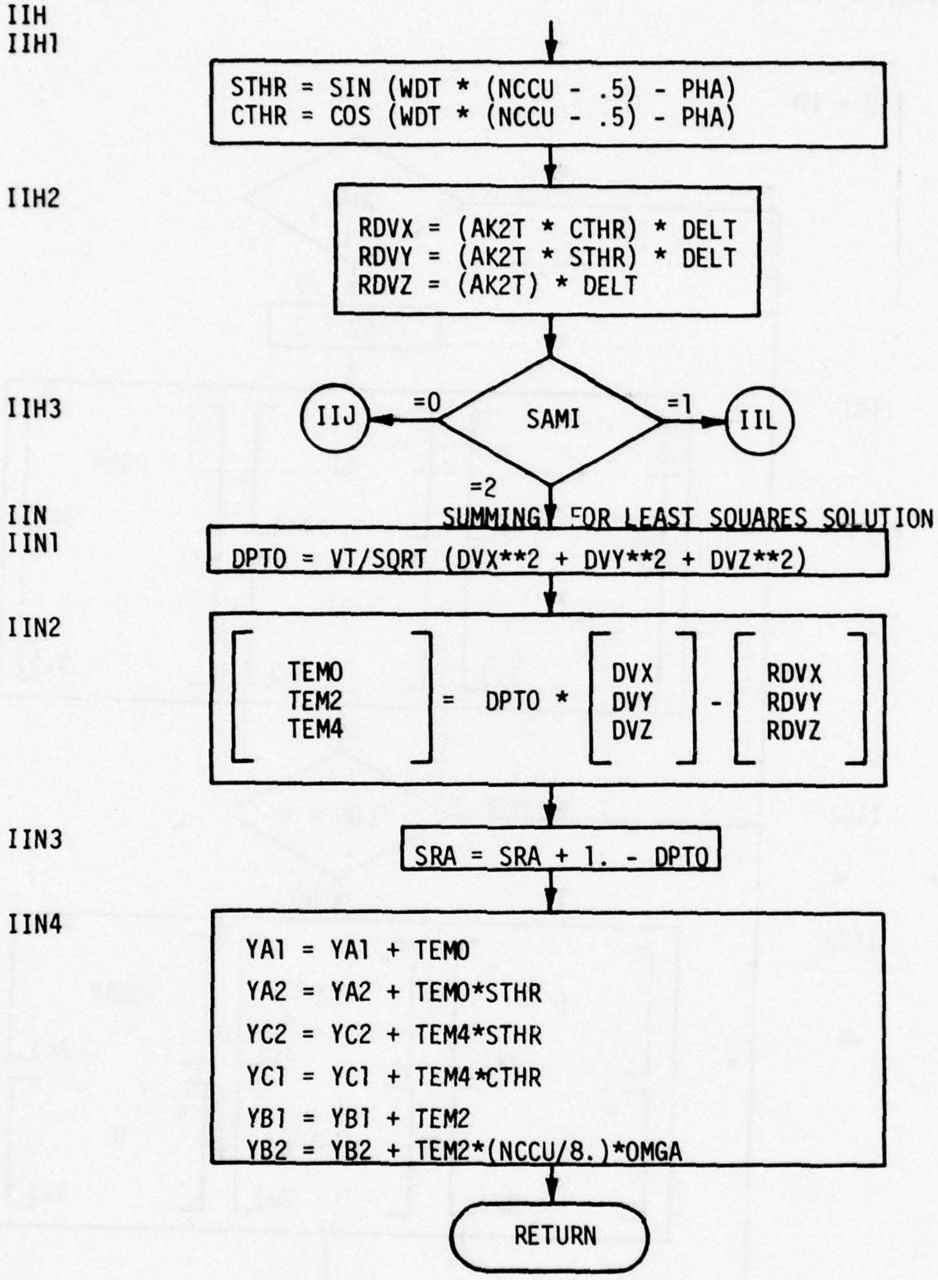
$$\left. + \begin{bmatrix} F1*CD33 & F1*CD34 & F1*CD35 \\ -F2*CD39 & -F2*CD40 & -F2*CD41 \\ F1*CD36 & F1*CD37 & F1*CD38 \end{bmatrix} \begin{bmatrix} DVI \\ DVJ \\ DVK \end{bmatrix} \right\}$$

G & SPEED DEPENDENT

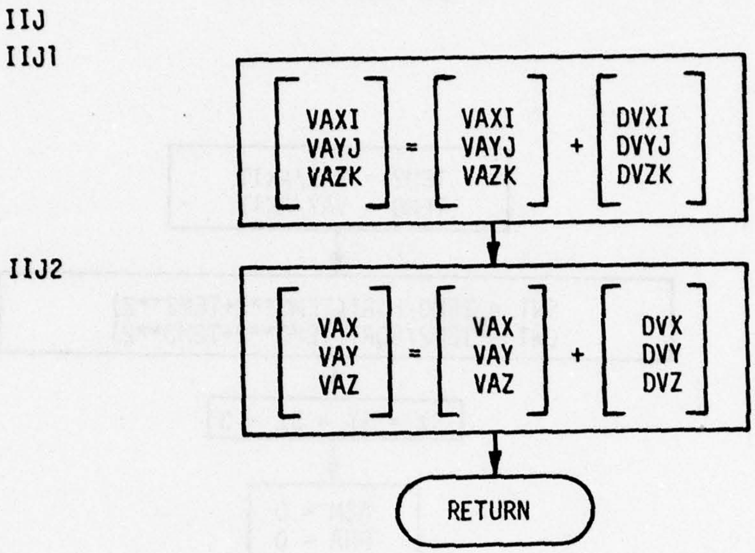
IIF13

$$\begin{bmatrix} DCAR \\ \quad 3x3 \end{bmatrix} = \begin{bmatrix} 0 & Q_z & Ø_y \\ -Ø_z & 0 & -Ø_x \\ -Ø_y & Q_x & 0 \end{bmatrix}$$

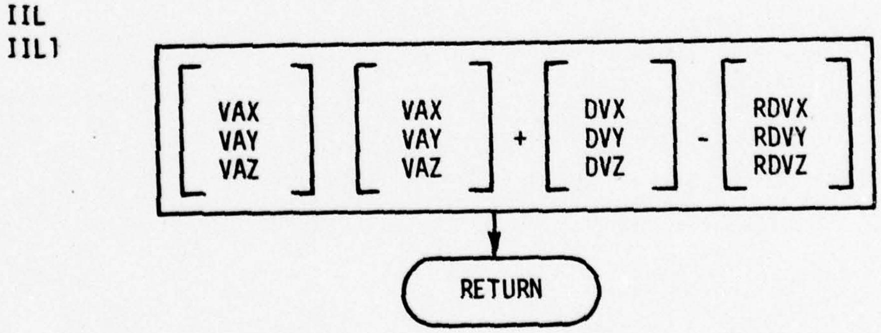

IIG = IM
CHAJ
=0
≠0
CHAJ = 0
IIG1
D 3x3 = AJ 3x3 DCAR 3x3
AJ 3x3 = AJ 3x3 + D 3x3
IIG2
FLGN = 0
YES
NO
IIG3
D 3x3 = SA 3x3 DCAR 3x3
SA 3x3 = SA 3x3 + D 3x3
RETURN


IIH
IIH1
STHR = SIN (WDT * (NCCU - .5) - PHA)
CTHR = COS (WDT * (NCCU - .5) - PHA)
IIH2
RDVX = (AK2T * CTHR) * DELT
RDVY = (AK2T * STHR) * DELT
RDVZ = (AK2T) * DELT
IIH3
IIJ
=0
SAMI
=1
IIL
=2
IIN
SUMMING FOR LEAST SQUARES SOLUTION
IIN1
DPTO = VT/SQRT (DVX**2 + DVY**2 + DVZ**2)
IIN2
TEM0
TEM2
TEM4
= DPTO *
DVX
DVY
DVZ
-
RDVX
RDVY
RDVZ
IIN3
SRA = SRA + 1. - DPTO
IIN4
YA1 = YA1 + TEM0
YA2 = YA2 + TEM0*STHR
YC2 = YC2 + TEM4*STHR
YC1 = YC1 + TEM4*CTHR
YB1 = YB1 + TEM2
YB2 = YB2 + TEM2*(NCCU/8.)*OMGA
RETURN

SUMMING FOR E.P.A. SOLUTION

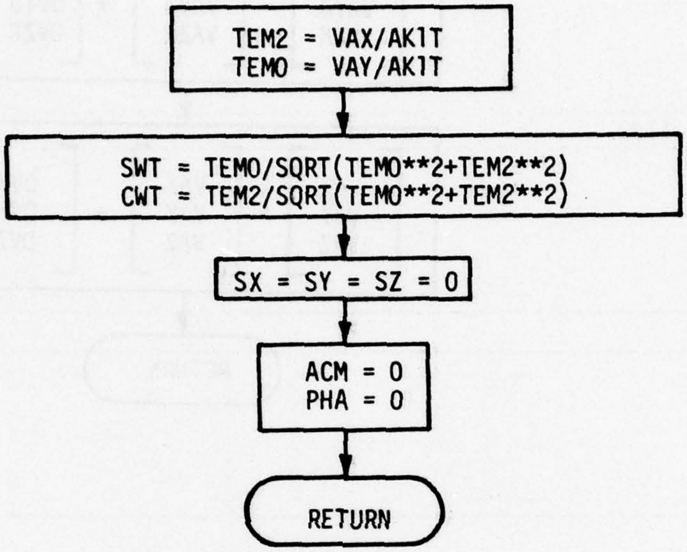

IIJ
IIJ1
$\begin{bmatrix} VAXI \\ VAYJ \\ VAZK \end{bmatrix} = \begin{bmatrix} VAXI \\ VAYJ \\ VAZK \end{bmatrix} + \begin{bmatrix} DVXI \\ DVYJ \\ DVZK \end{bmatrix}$
IIJ2
$\begin{bmatrix} VAX \\ VAY \\ VAZ \end{bmatrix} = \begin{bmatrix} VAX \\ VAY \\ VAZ \end{bmatrix} + \begin{bmatrix} DVX \\ DVY \\ DVZ \end{bmatrix}$
RETURN

SUMMING FOR LOCAL LEVEL SOLUTION


IIL
IIL1
$\begin{bmatrix} VAX \\ VAY \\ VAZ \end{bmatrix} \begin{bmatrix} VAX \\ VAY \\ VAZ \end{bmatrix} + \begin{bmatrix} DVX \\ DVY \\ DVZ \end{bmatrix} - \begin{bmatrix} RDVX \\ RDVY \\ RDVZ \end{bmatrix}$
RETURN

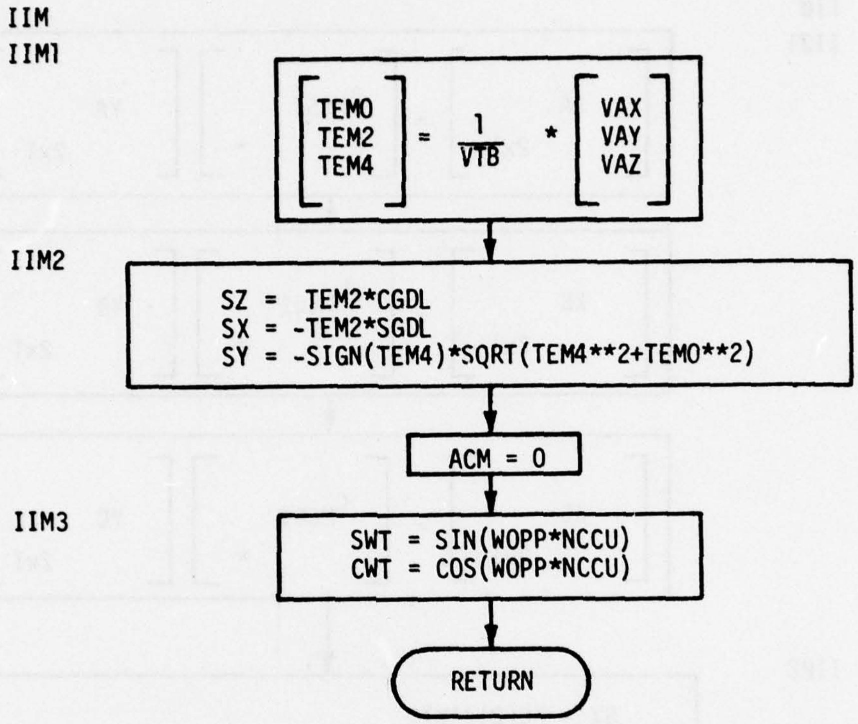
E.P.A. SOLUTION

IIK
IIK2



```
TEM2 = VAX/AK1T
TEMO = VAY/AK1T
SWT = TEMO/SQRT(TEMO**2+TEM2**2)
CWT = TEM2/SQRT(TEMO**2+TEM2**2)
SX = SY = SZ = 0
ACM = 0
PHA = 0
```

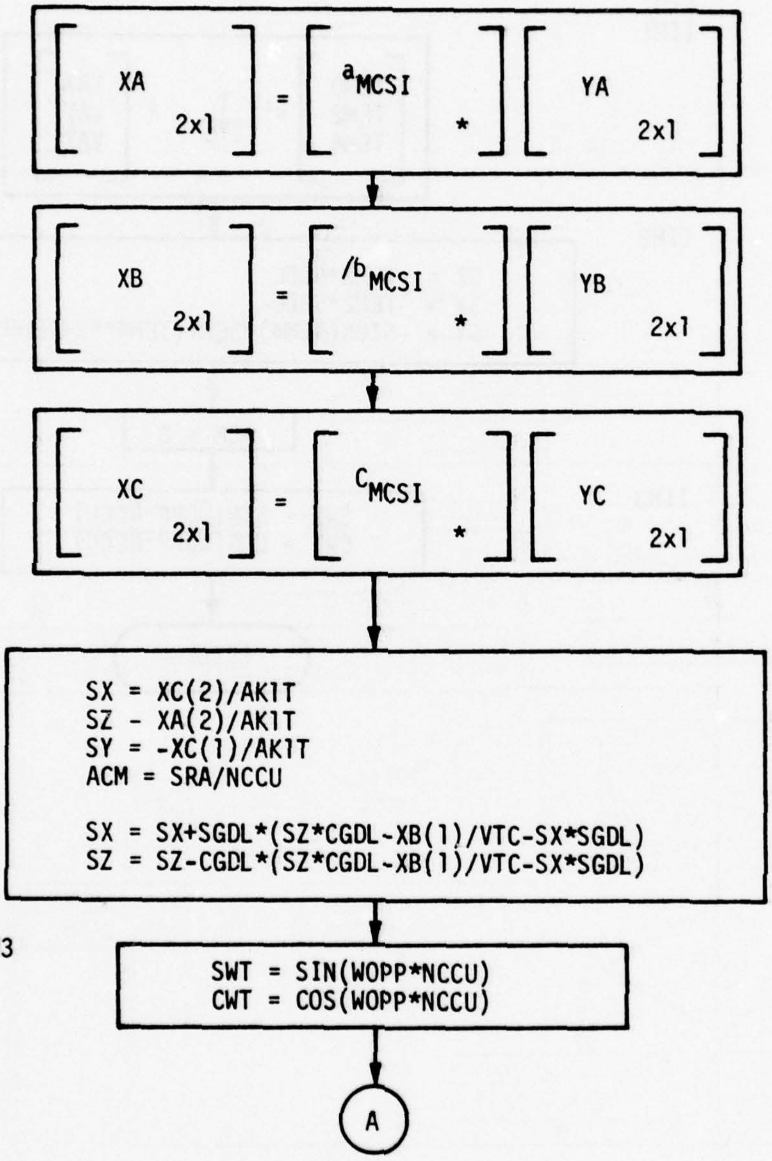
LOCAL LEVEL SOLUTION



LEAST SQUARES SOLUTION

II∅

II∅1



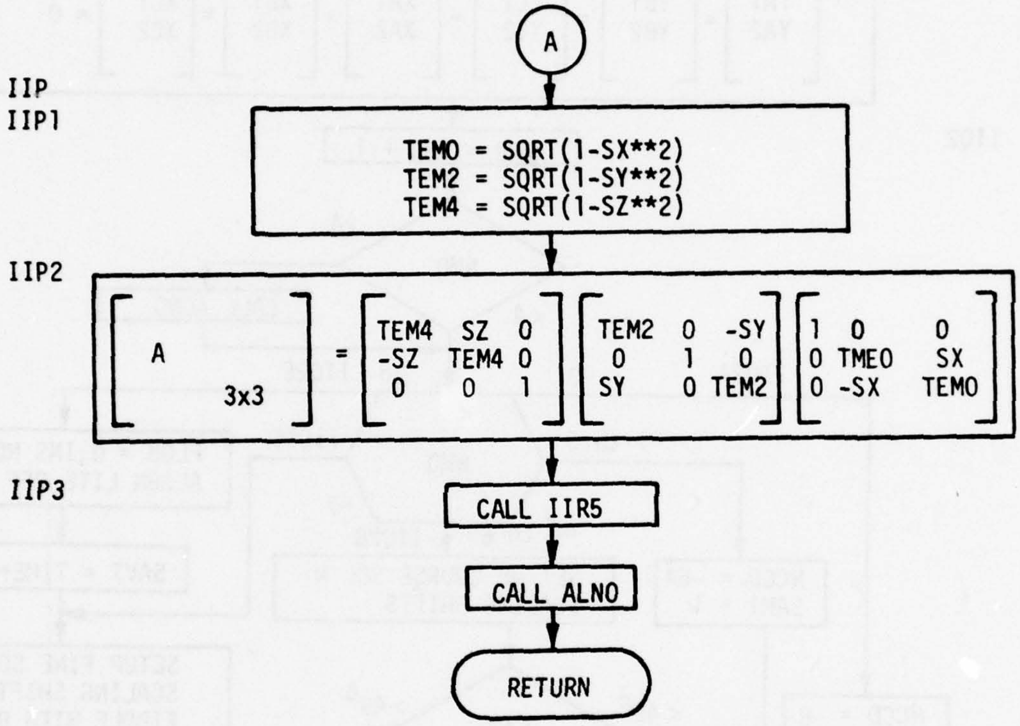
II∅2

```
SX = XC(2)/AK1T
SZ - XA(2)/AK1T
SY = -XC(1)/AK1T
ACM = SRA/NCCU

SX = SX+SGDL*(SZ*CGDL-XB(1)/VTC-SX*SGDL)
SZ = SZ-CGDL*(SZ*CGDL-XB(1)/VTC-SX*SGDL)
```

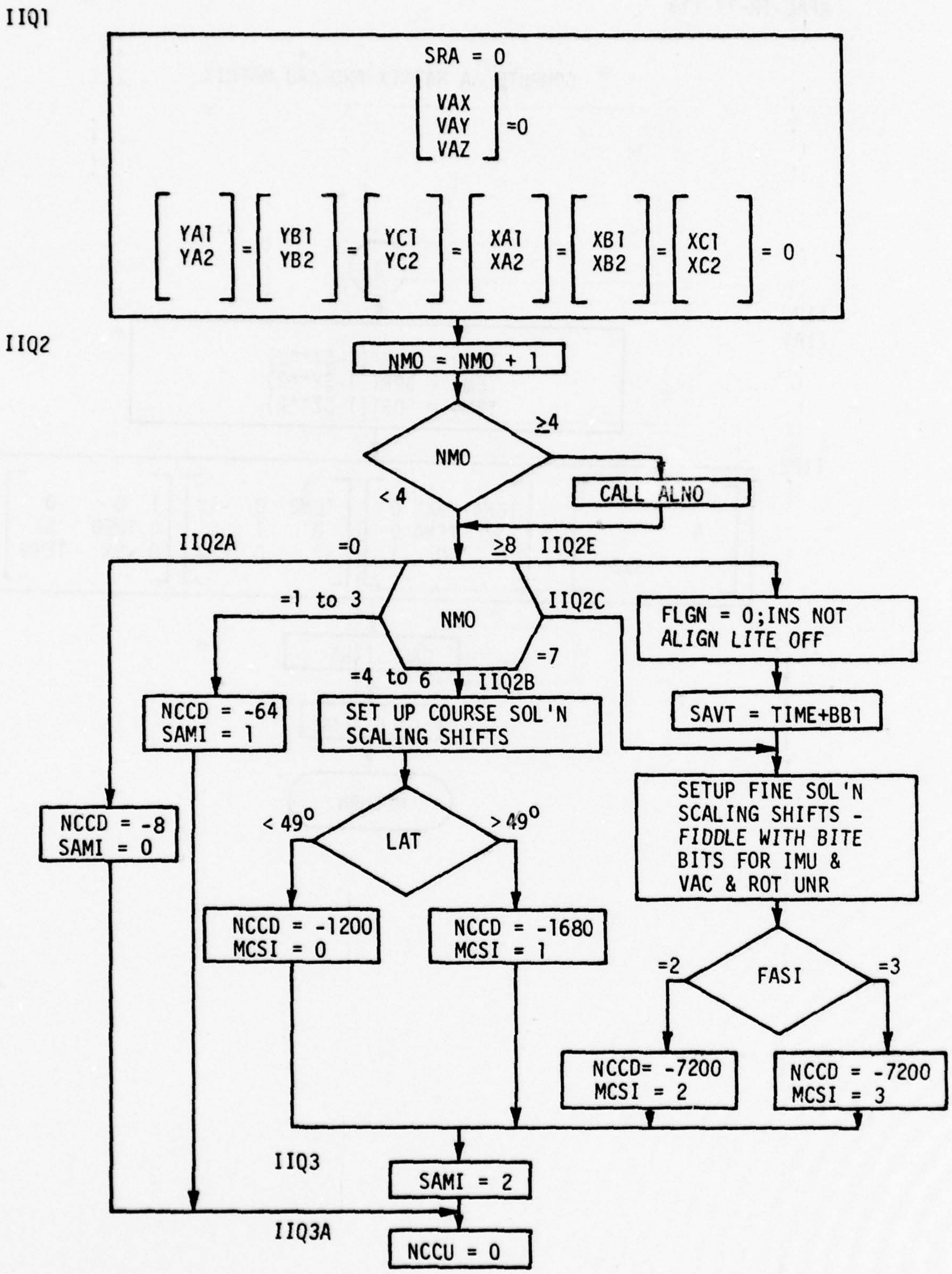
II04 = IIM3

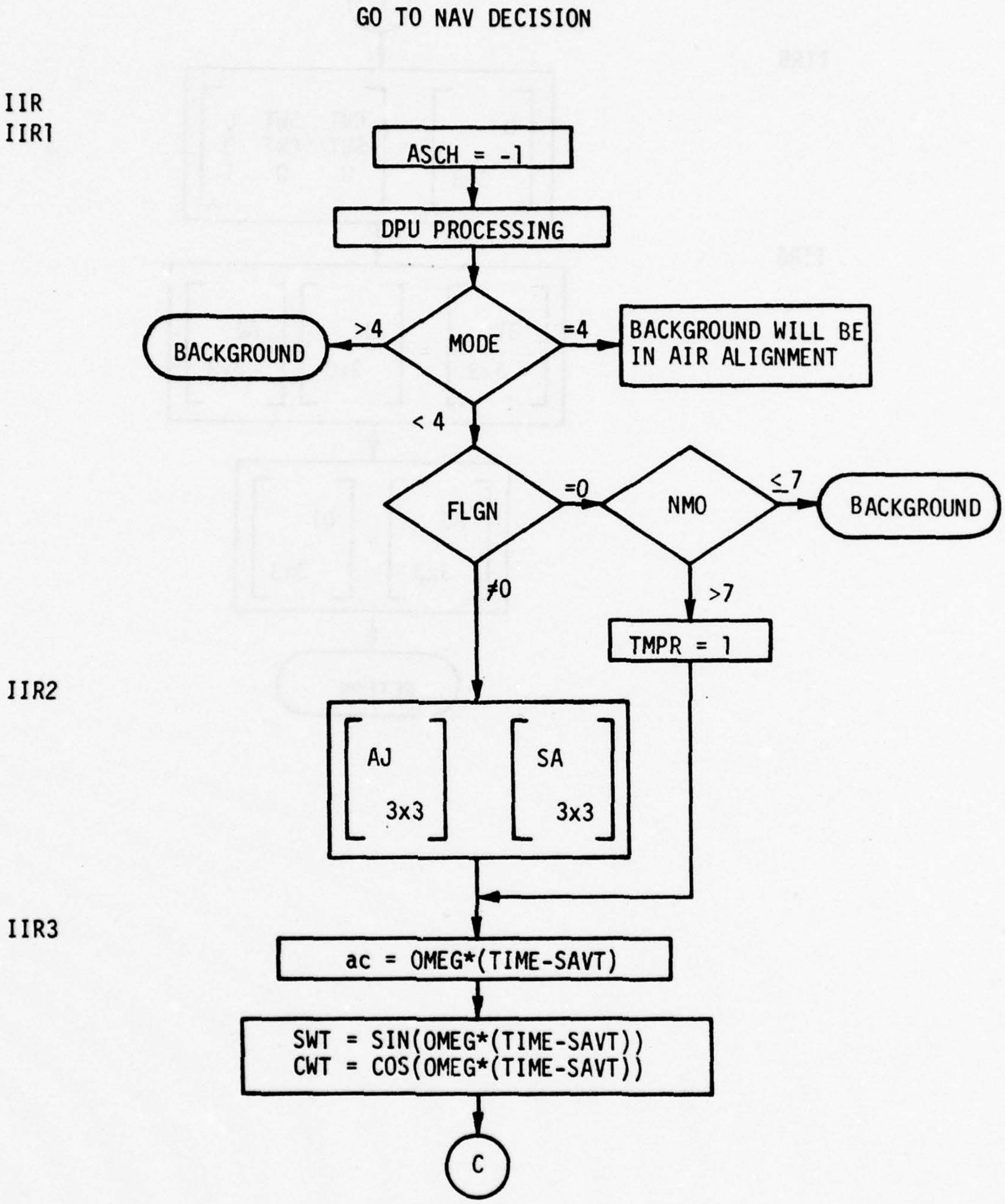
```
SWT = SIN(WOPP*NCCU)
CWT = COS(WOPP*NCCU)
```

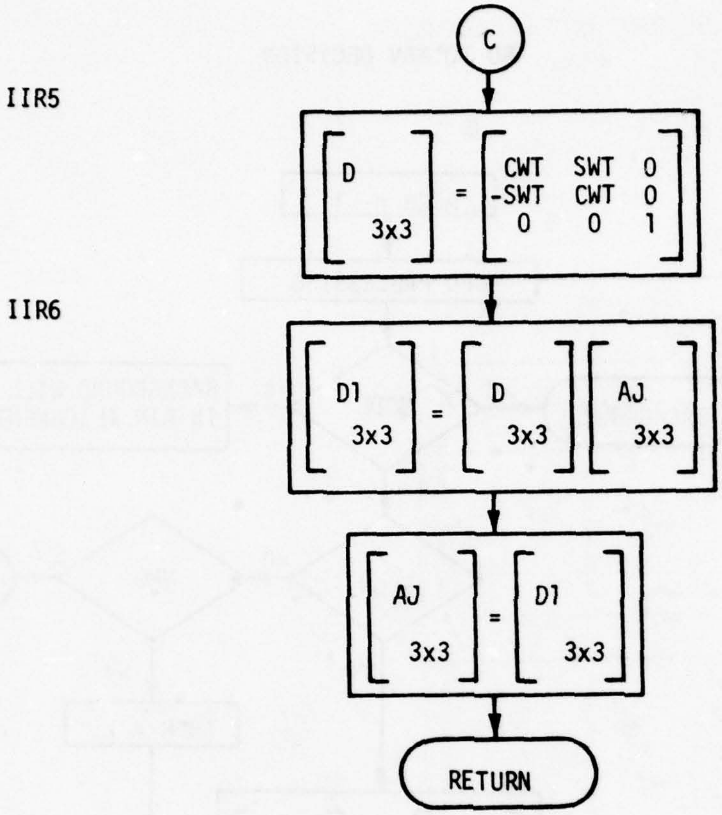
COMPUTE ΔA MATRIX AND ΔAJ MATRIX


A
IIP
IIP1
TEMO = SQRT(1-SX**2)
TEM2 = SQRT(1-SY**2)
TEM4 = SQRT(1-SZ**2)
IIP2
A 3x3 = [TEM4 SZ 0; -SZ TEM4 0; 0 0 1] [TEM2 0 -SY; 0 1 0; SY 0 TEM2] [1 0 0; 0 TMEO SX; 0 -SX TEMO]
IIP3
CALL IIR5
CALL ALNO
RETURN

SUBROUTINE RESET

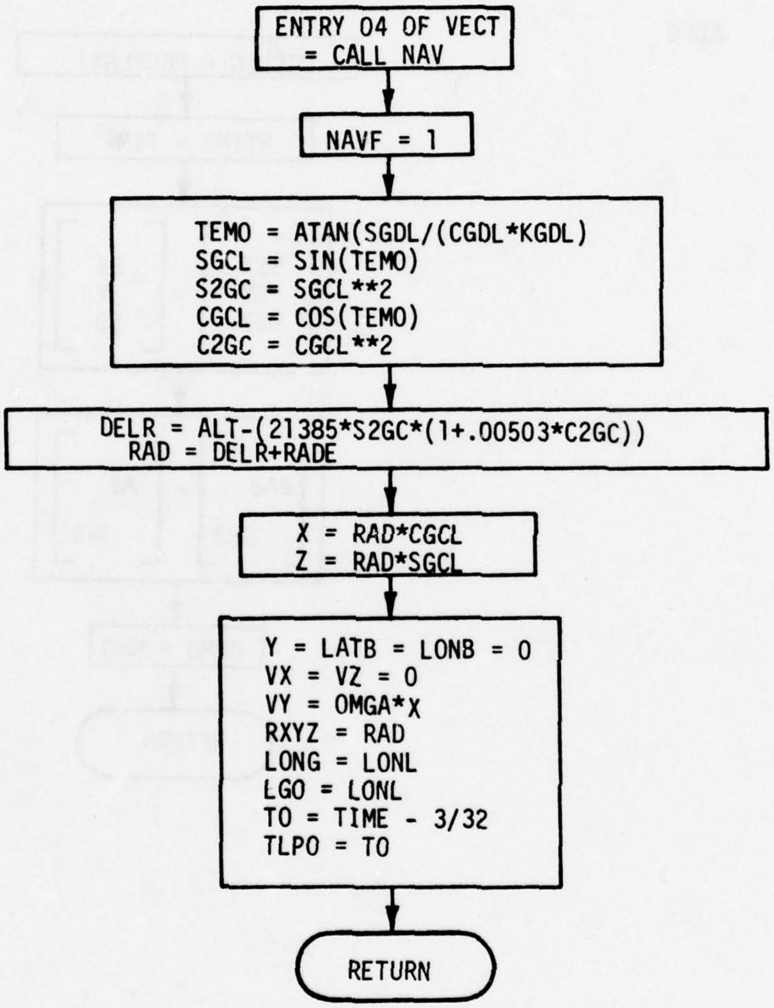



GO TO NAV DECISION
IIR
IIR1
ASCH = -1
DPU PROCESSING
MODE
>4
BACKGROUND
=4
BACKGROUND WILL BE IN AIR ALIGNMENT
< 4
FLGN
=0
NMO
≤ 7
BACKGROUND
>7
TMPR = 1
≠0
IIR2
AJ 3x3
SA 3x3
IIR3
ac = OMEG*(TIME-SAVT)
SWT = SIN(OMEG*(TIME-SAVT))
CWT = COS(OMEG*(TIME-SAVT))
C


C
IIR5
D 3x3 = CWT SWT 0 / -SWT CWT 0 / 0 0 1
IIR6
D1 3x3 = D 3x3 AJ 3x3
AJ 3x3 = D1 3x3
RETURN

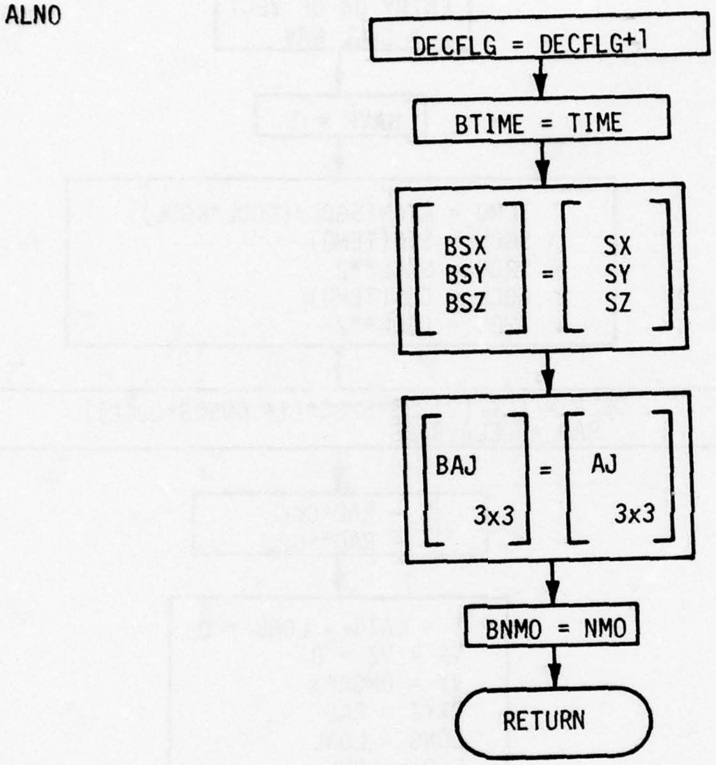
NAVIGATION INITIALIZATION

NAVI

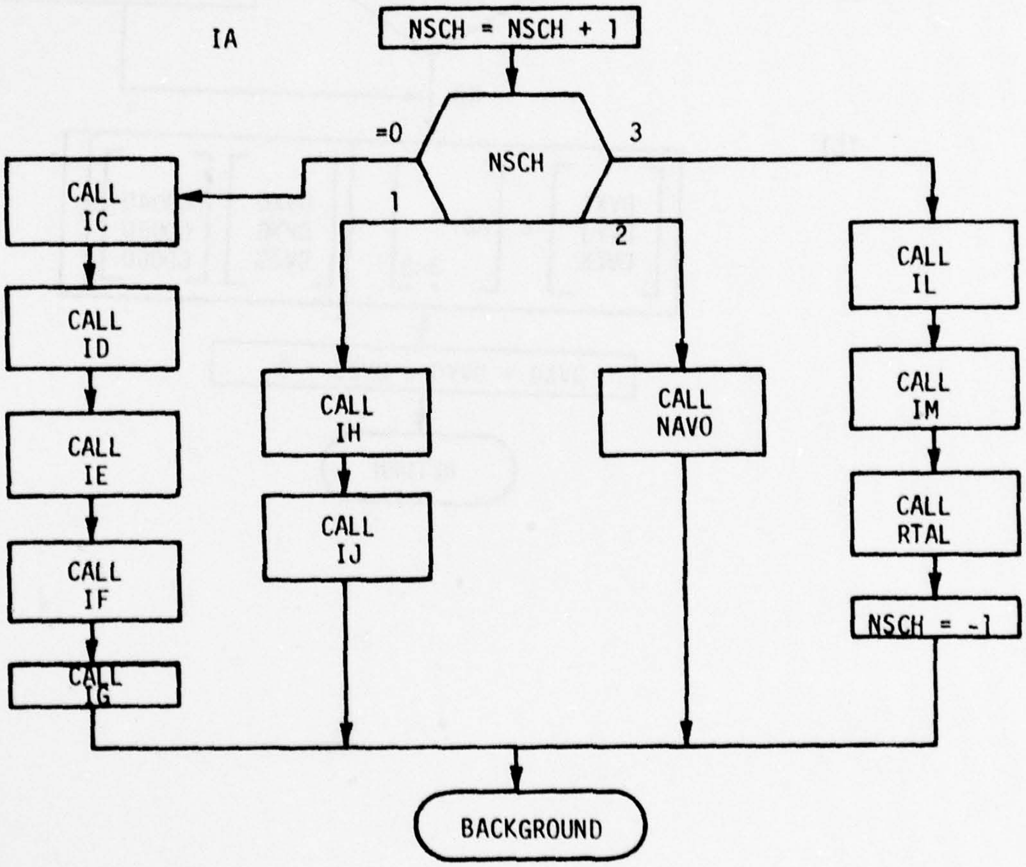


```
ENTRY 04 OF VECT
= CALL NAV

NAVF = 1

TEMO = ATAN(SGDL/(CGDL*KGDL)
SGCL = SIN(TEMO)
S2GC = SGCL**2
CGCL = COS(TEMO)
C2GC = CGCL**2

DELR = ALT-(21385*S2GC*(1+.00503*C2GC))
  RAD = DELR+RADE

X = RAD*CGCL
Z = RAD*SGCL

Y = LATB = LONB = 0
VX = VZ = 0
VY = OMGA*X
RXYZ = RAD
LONG = LONL
LGO = LONL
TO = TIME - 3/32
TLPO = TO

RETURN
```

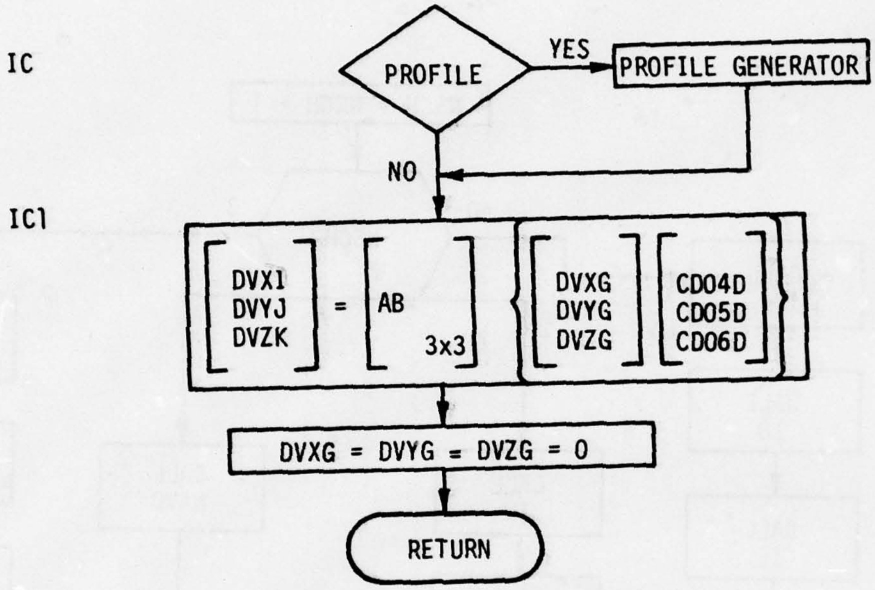
ALIGNMENT OUTPUT ROUTINE



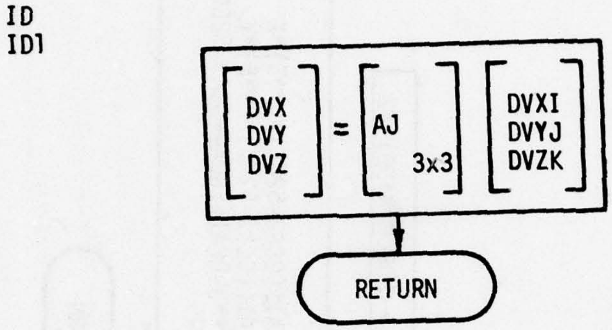
NAVIGATION SUB-EXECUTIVE



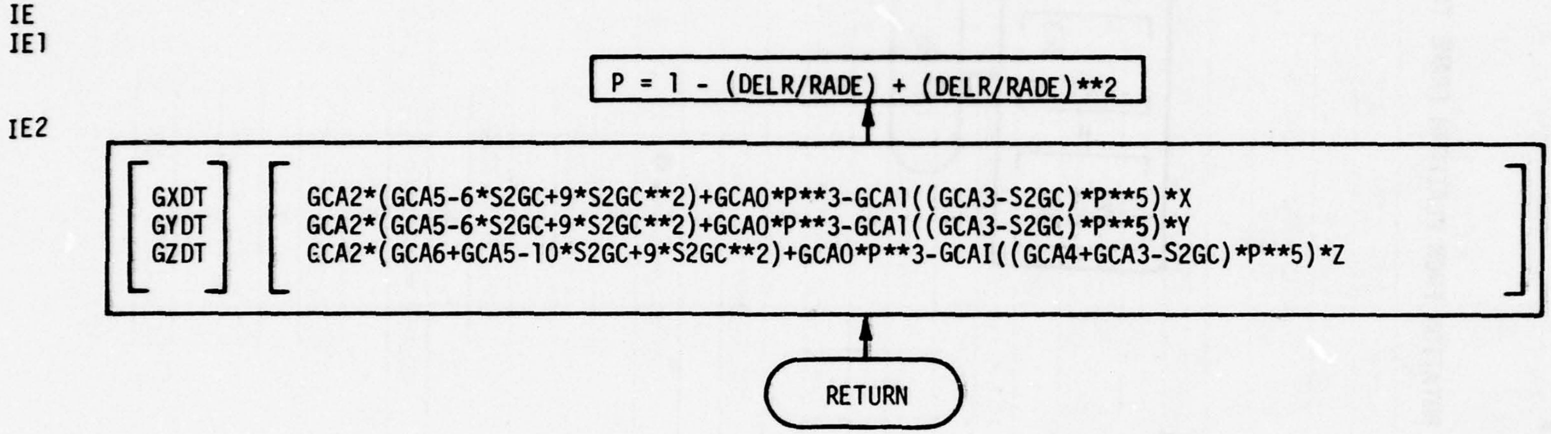
ACCELEROMETER BIAS & SCALE FACTOR COMPUTATION
AND NON-ORTHOGONALITY COMPENSATION



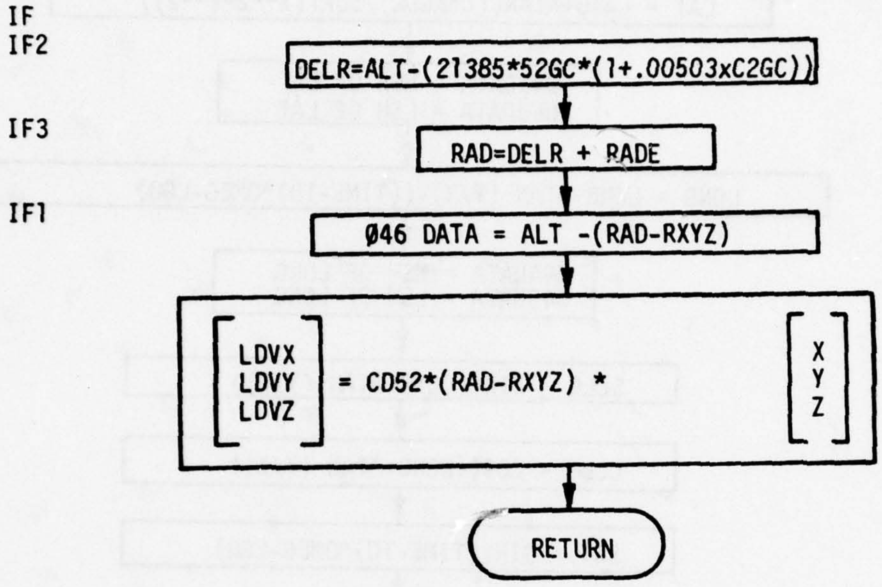
ROTATION FROM PLATFORM FRAME TO NAVIGATION FRAME



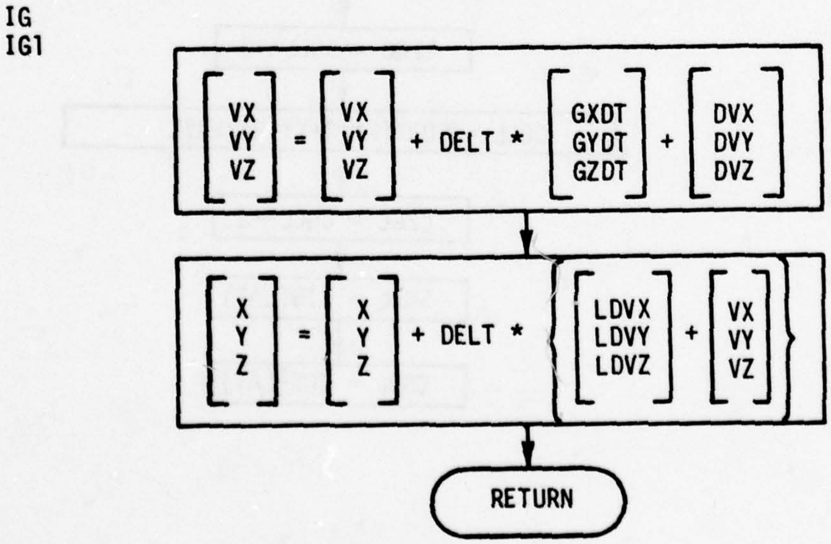
GRAVITY MODEL


IE
IE1
IE2
P = 1 - (DELR/RADE) + (DELR/RADE)**2
GXDT
GYDT
GZDT
GCA2*(GCA5-6*S2GC+9*S2GC**2)+GCA0*P**3-GCA1((GCA3-S2GC)*P**5)*X
GCA2*(GCA5-6*S2GC+9*S2GC**2)+GCA0*P**3-GCA1((GCA3-S2GC)*P**5)*Y
GCA2*(GCA6+GCA5-10*S2GC+9*S2GC**2)+GCA0*P**3-GCAI((GCA4+GCA3-S2GC)*P**5)*Z
RETURN

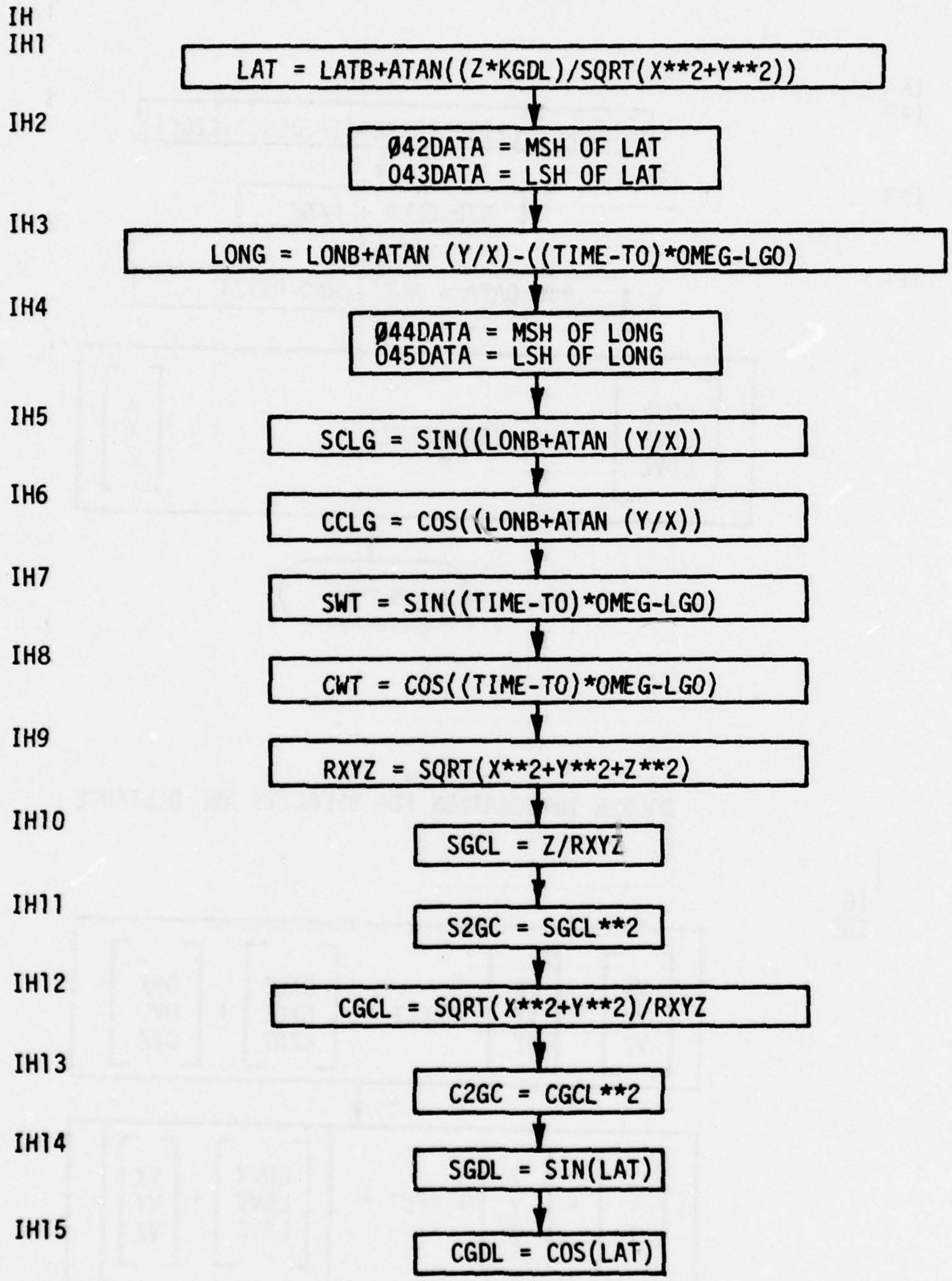
VERTICAL DAMPING COMPUTATION



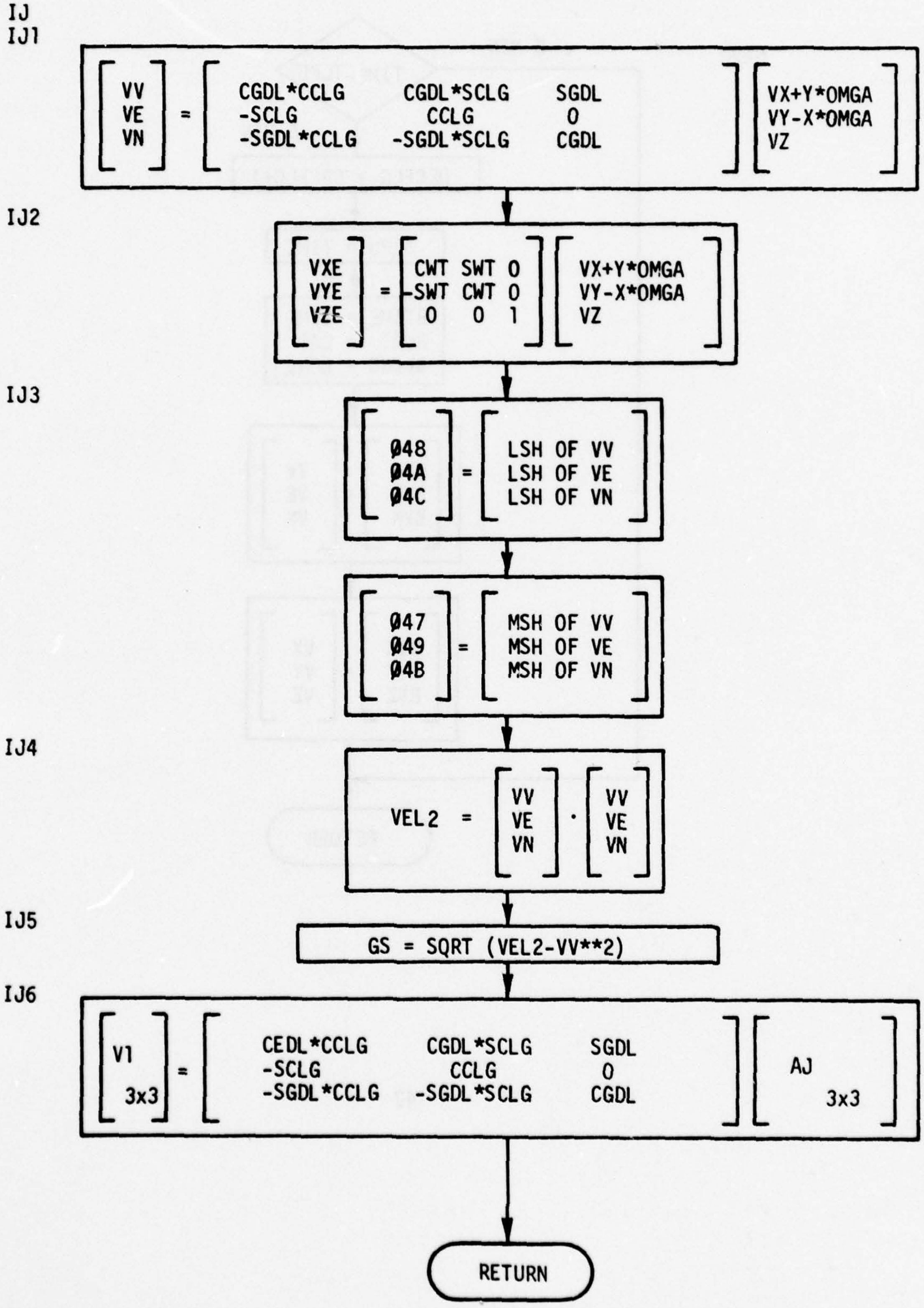
DOUBLE INTEGRATION FOR VELOCITY AND DISTANCE



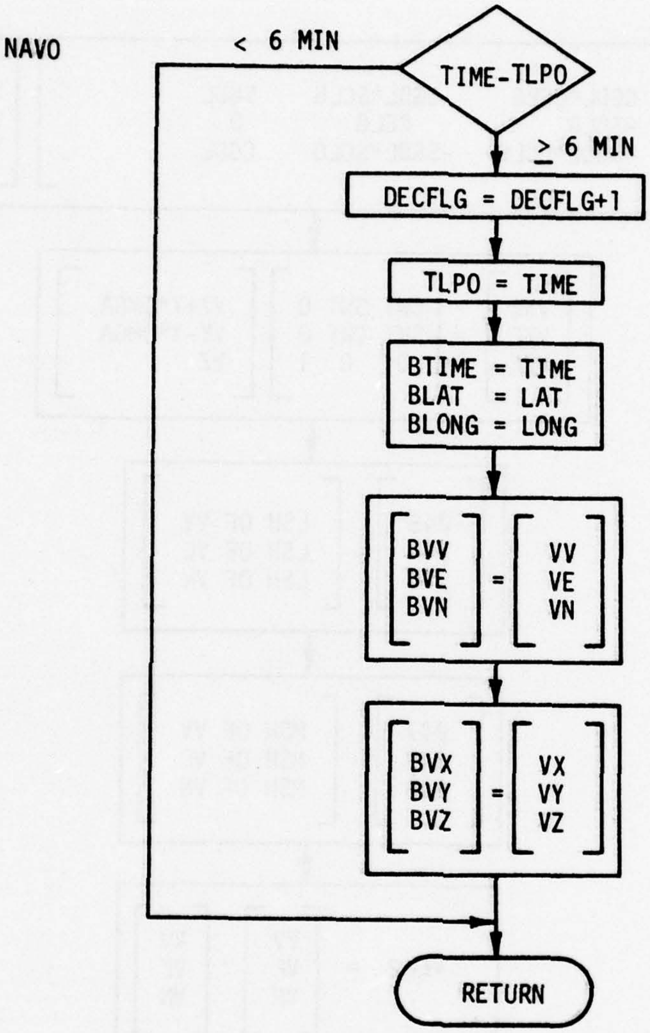
LATITUDE AND LONGITUDE COMPUTATION


IH
IH1
LAT = LATB+ATAN((Z*KGDL)/SQRT(X**2+Y**2))
IH2
Ø42DATA = MSH OF LAT
043DATA = LSH OF LAT
IH3
LONG = LONB+ATAN (Y/X)-((TIME-TO)*OMEG-LGO)
IH4
Ø44DATA = MSH OF LONG
045DATA = LSH OF LONG
IH5
SCLG = SIN((LONB+ATAN (Y/X))
IH6
CCLG = COS((LONB+ATAN (Y/X))
IH7
SWT = SIN((TIME-TO)*OMEG-LGO)
IH8
CWT = COS((TIME-TO)*OMEG-LGO)
IH9
RXYZ = SQRT(X**2+Y**2+Z**2)
IH10
SGCL = Z/RXYZ
IH11
S2GC = SGCL**2
IH12
CGCL = SQRT(X**2+Y**2)/RXYZ
IH13
C2GC = CGCL**2
IH14
SGDL = SIN(LAT)
IH15
CGDL = COS(LAT)

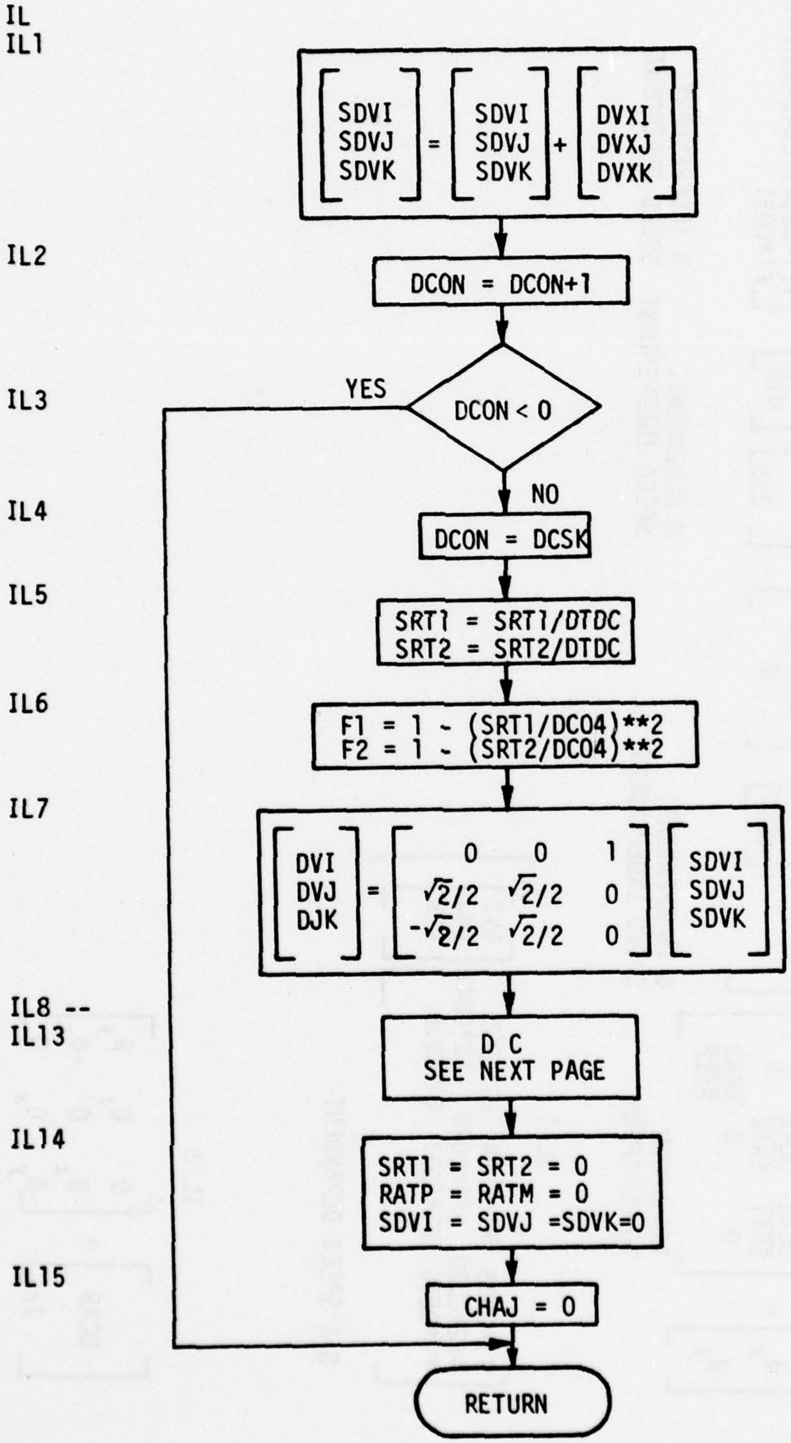
LOCAL VERTICAL CO-ORDINATES AND GROUND SPEED


IJ
IJ1
VV VE VN = CGDL*CCLG CGDL*SCLG SGDL -SCLG CCLG 0 -SGDL*CCLG -SGDL*SCLG CGDL VX+Y*OMGA VY-X*OMGA VZ
IJ2
VXE VYE VZE = CWT SWT 0 -SWT CWT 0 0 0 1 VX+Y*OMGA VY-X*OMGA VZ
IJ3
Ø48 Ø4A Ø4C = LSH OF VV LSH OF VE LSH OF VN
Ø47 Ø49 Ø4B = MSH OF VV MSH OF VE MSH OF VN
IJ4
VEL2 = VV VE VN · VV VE VN
IJ5
GS = SQRT (VEL2-VV**2)
IJ6
V1 3x3 = CEDL*CCLG CGDL*SCLG SGDL -SCLG CCLG 0 -SGDL*CCLG -SGDL*SCLG CGDL AJ 3x3
RETURN

NAVIGATION OUTPUT ROUTINE



DRIFT COMPENSATION



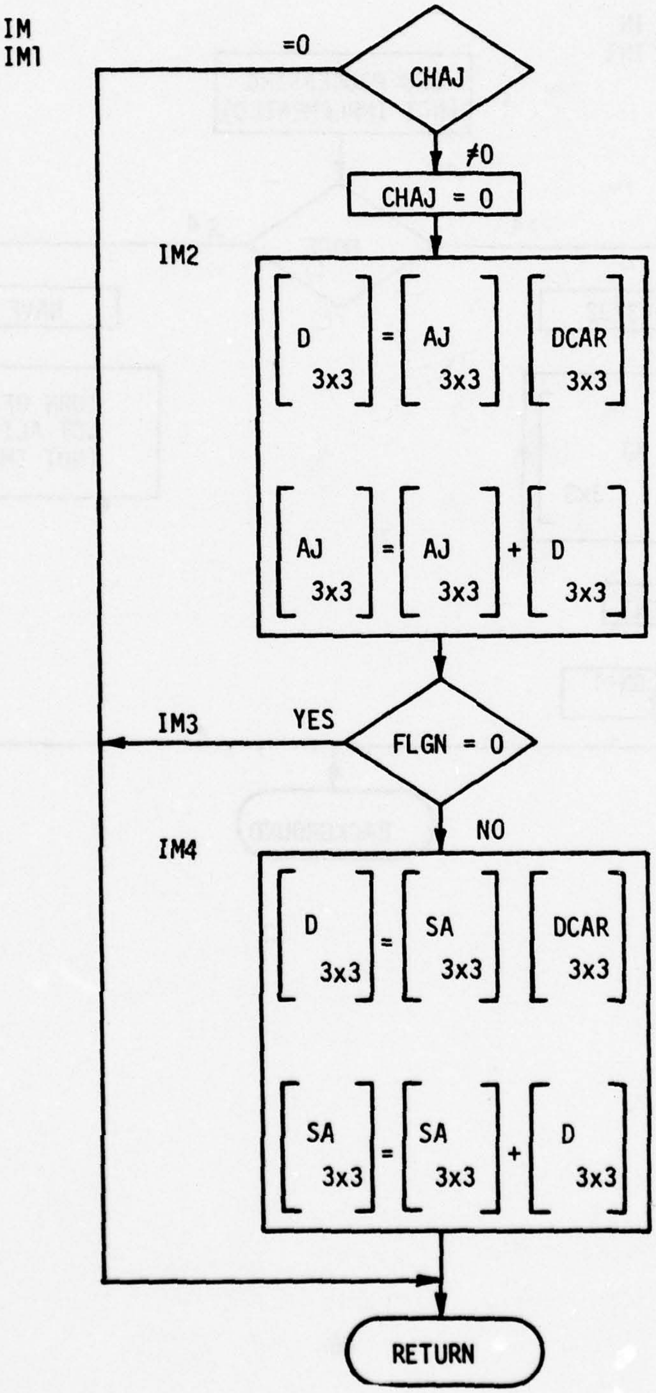
$$\begin{bmatrix} \emptyset_x \\ \emptyset_y \\ \emptyset_z \end{bmatrix} = \underset{\text{TRANSFORM TO PLATFORM}}{\overset{\text{IL12}}{\begin{bmatrix} \frac{DC42}{SRT1} & \frac{DC42}{SRT2} & 0 \\ \frac{DC42-}{SRT1} & \frac{DC42}{SRT2} & 0 \\ 0 & 0 & \frac{DC43}{SRT2} \end{bmatrix}}} \left\{ \underset{\text{G INDEPENDENT, SPEED INDEPENDENT}}{\overset{\text{IL8}}{DTDC * \begin{bmatrix} CD16 \\ CD18 \\ CD17 \end{bmatrix} + \begin{bmatrix} 0 \\ CD28*RATP \\ -CD29*RATM \\ 0 \end{bmatrix}}} + \underset{\text{G DEPENDENT, SPEED INDEPENDENT}}{\overset{\text{IL9}}{\begin{bmatrix} GM \\ 3x3 \end{bmatrix} \begin{bmatrix} DVI \\ DVJ \\ DVK \end{bmatrix}}} + \underset{\text{G INDEPENDENT, SPEED DEPENDENT}}{\overset{\text{IL10}}{\begin{bmatrix} F1*CD30 \\ F2*CD32 \\ SRT2 \\ +(1-DC04)(CD42*RATP-CD43*RATM) \\ F1*CD31 \end{bmatrix}}} + \right.$$

$$\left. \underset{\text{G \& SPEED DEPENDENT}}{\overset{\text{IL11}}{\begin{bmatrix} F1*CD33 & F1*CD34 & F1*CD35 \\ -F2*CD39 & -F2*CD40 & -F2*CD41 \\ F1*CD36 & F1*CD37 & F1*CD38 \end{bmatrix} \begin{bmatrix} DVI \\ DVJ \\ DVK \end{bmatrix}}} \right\}$$

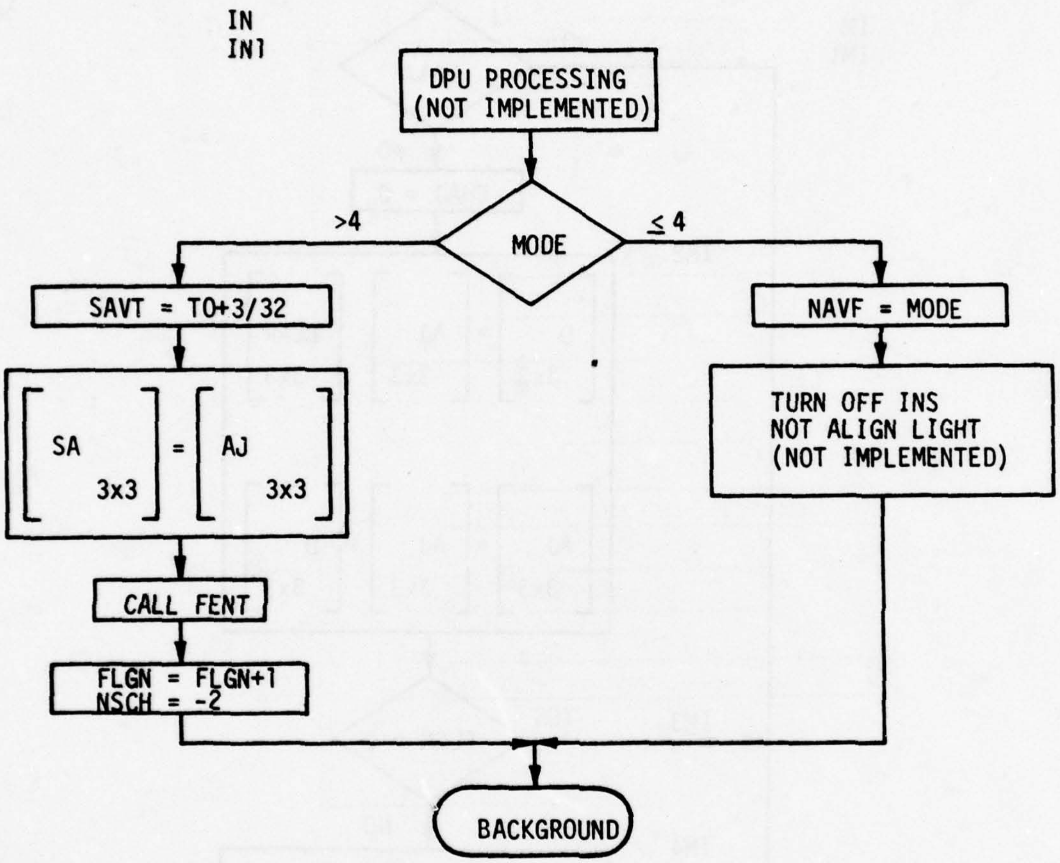
IL13

$$\begin{bmatrix} DCAR \\ 3x3 \end{bmatrix} = \begin{bmatrix} 0 & Q_z & \emptyset_y \\ -\emptyset_z & 0 & -\emptyset_x \\ -\emptyset_y & Q_x & 0 \end{bmatrix}$$

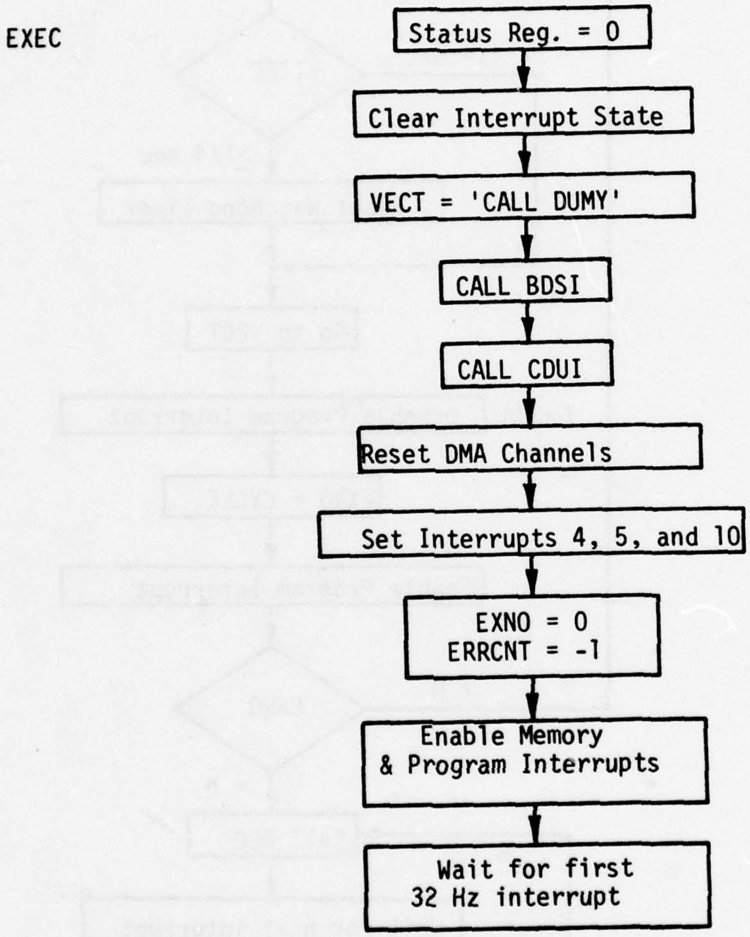
UPDATE AJ AND SA MATRICES



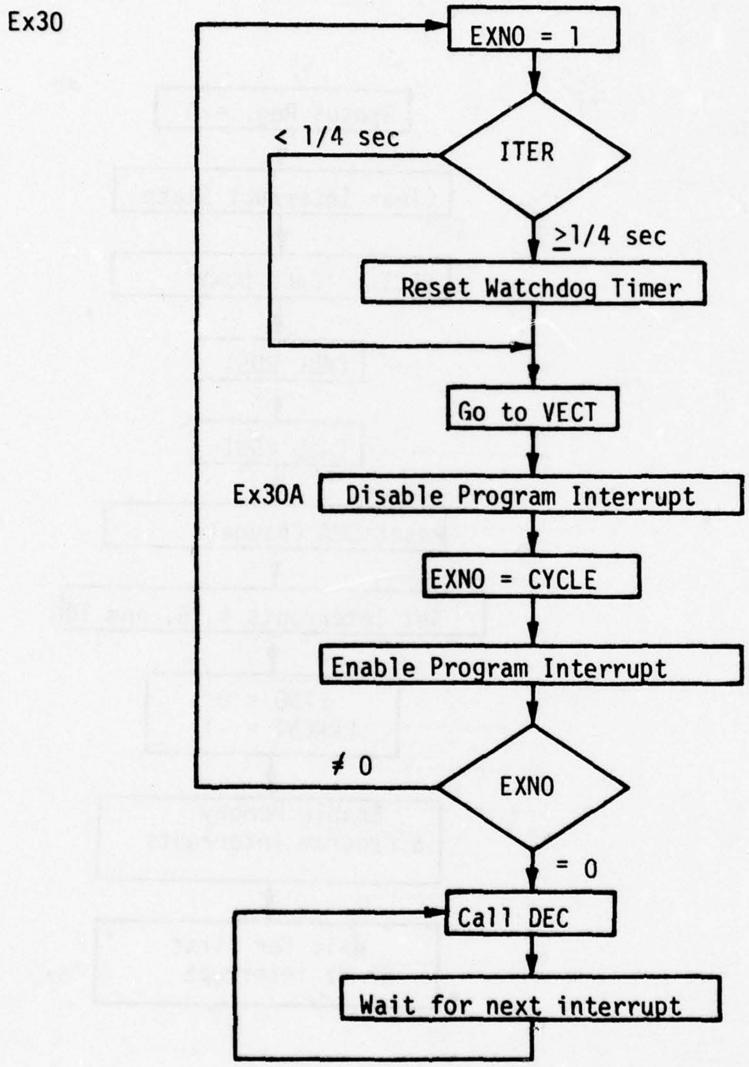
RETURN TO ALIGN DECISION (RTAL)



EXECUTIVE INITIALIZATION



SKC-2000 Executive



Vector Table

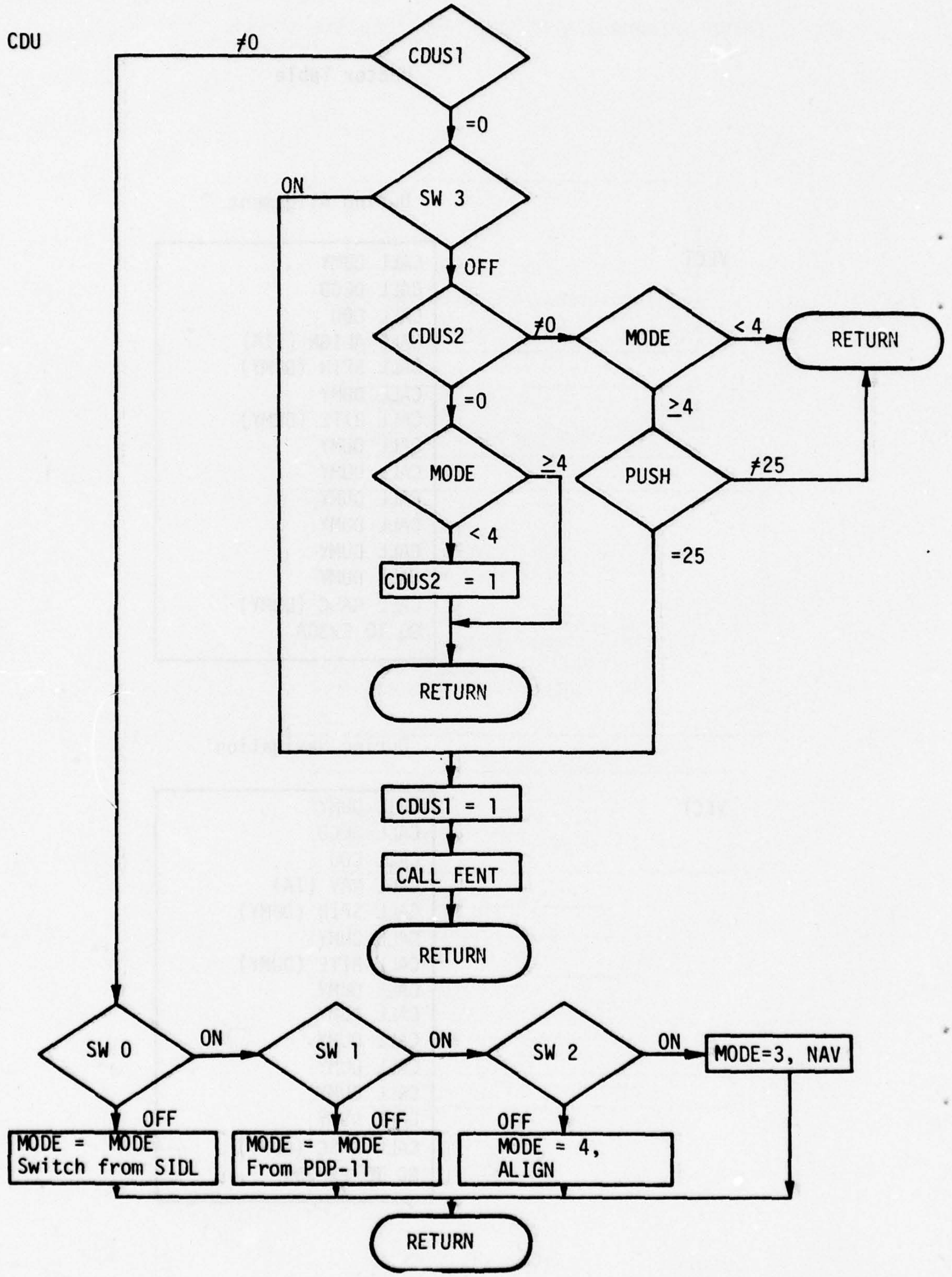
During Alignment

```
VECT    CALL DUMY
        CALL DECD
        CALL CDU
        CALL ALIGN (IIA)
        CALL SPIN (DUMY)
        CALL DUMY
        CALL BITE (DUMY)
        CALL DUMY
        CALL DUMY
        CALL DUMY
        CALL DUMY
        CALL DUMY
        CALL DUMY
        CALL GASC (DUMY)
        GO TO Ex30A
```

During Navigation

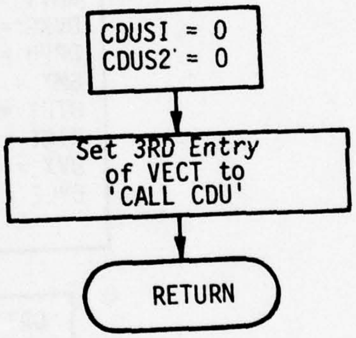
```
VECT    CALL DUMY
        CALL DECD
        CALL CDU
        CALL NAV (IA)
        CALL SPIN (DUMY)
        CALL DUMY
        CALL BITE (DUMY)
        CALL DUMY
        CALL DUMY
        CALL DUMY
        CALL DUMY
        CALL DUMY
        CALL DUMY
        CALL GASC (DUMY)
        GO TO Ex 30A
```

Syncronize SKC-2000 Alignment with Honeywell Alignment.



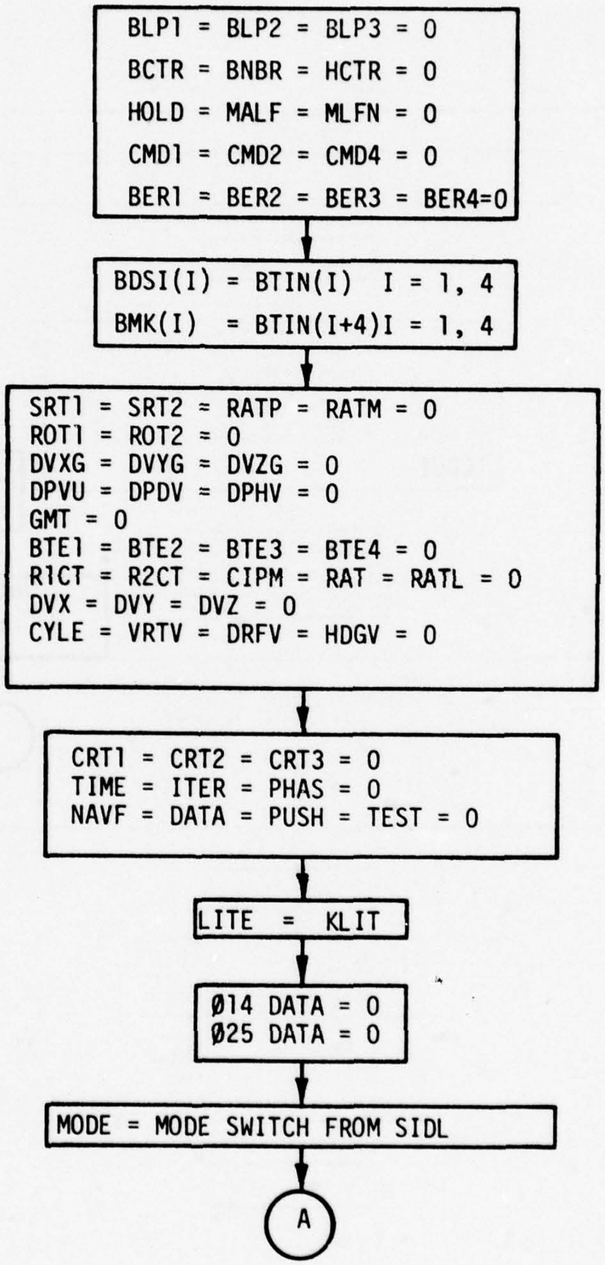
CDU Initialization

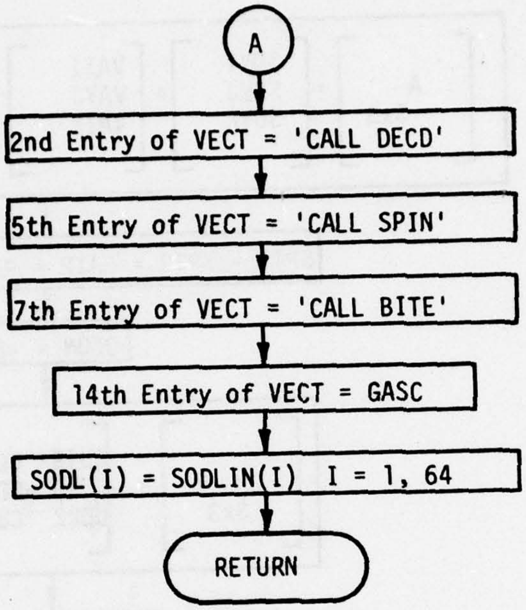
CDUI



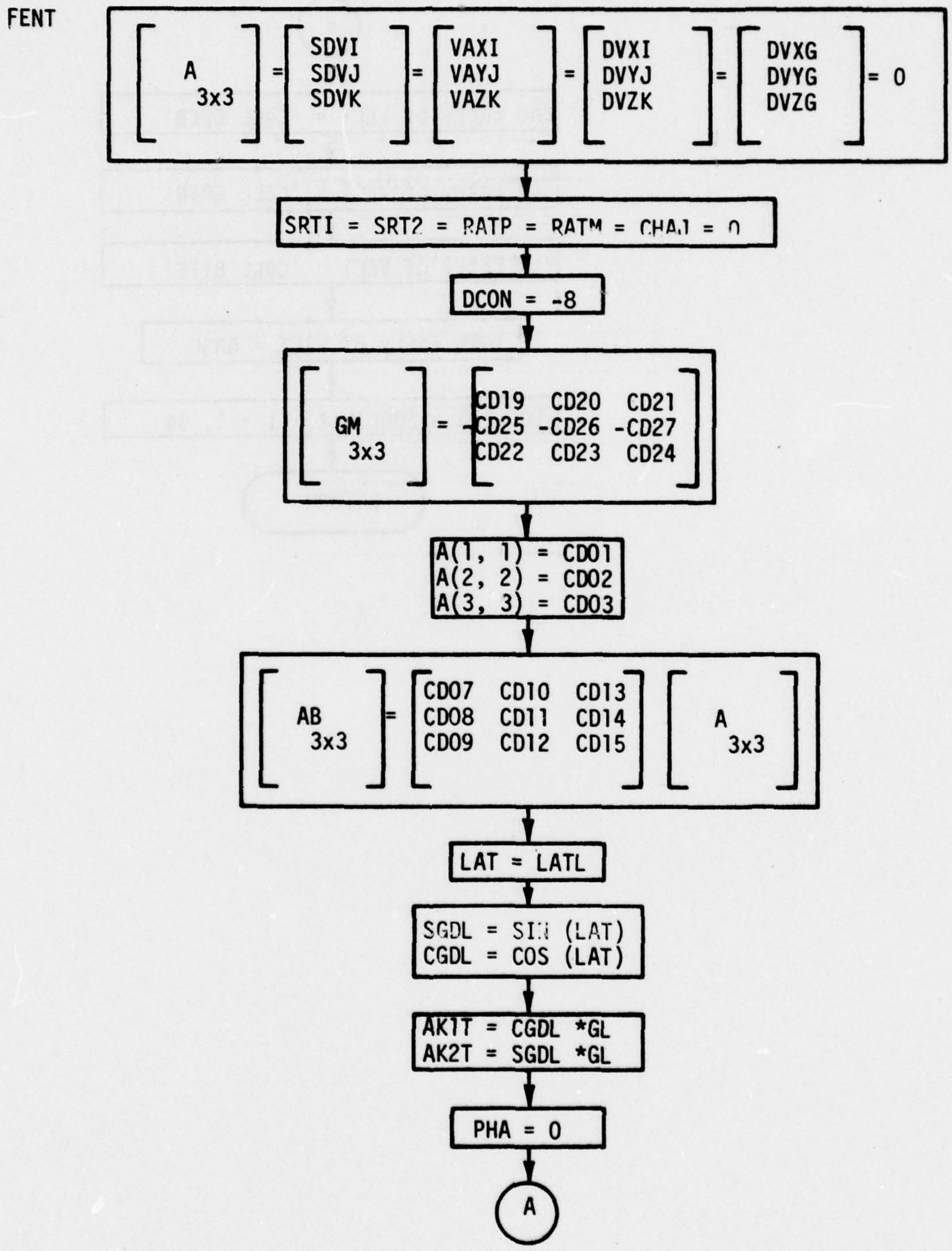
Initialize Built-In Test, Data Decode, & Auto Sequencing.

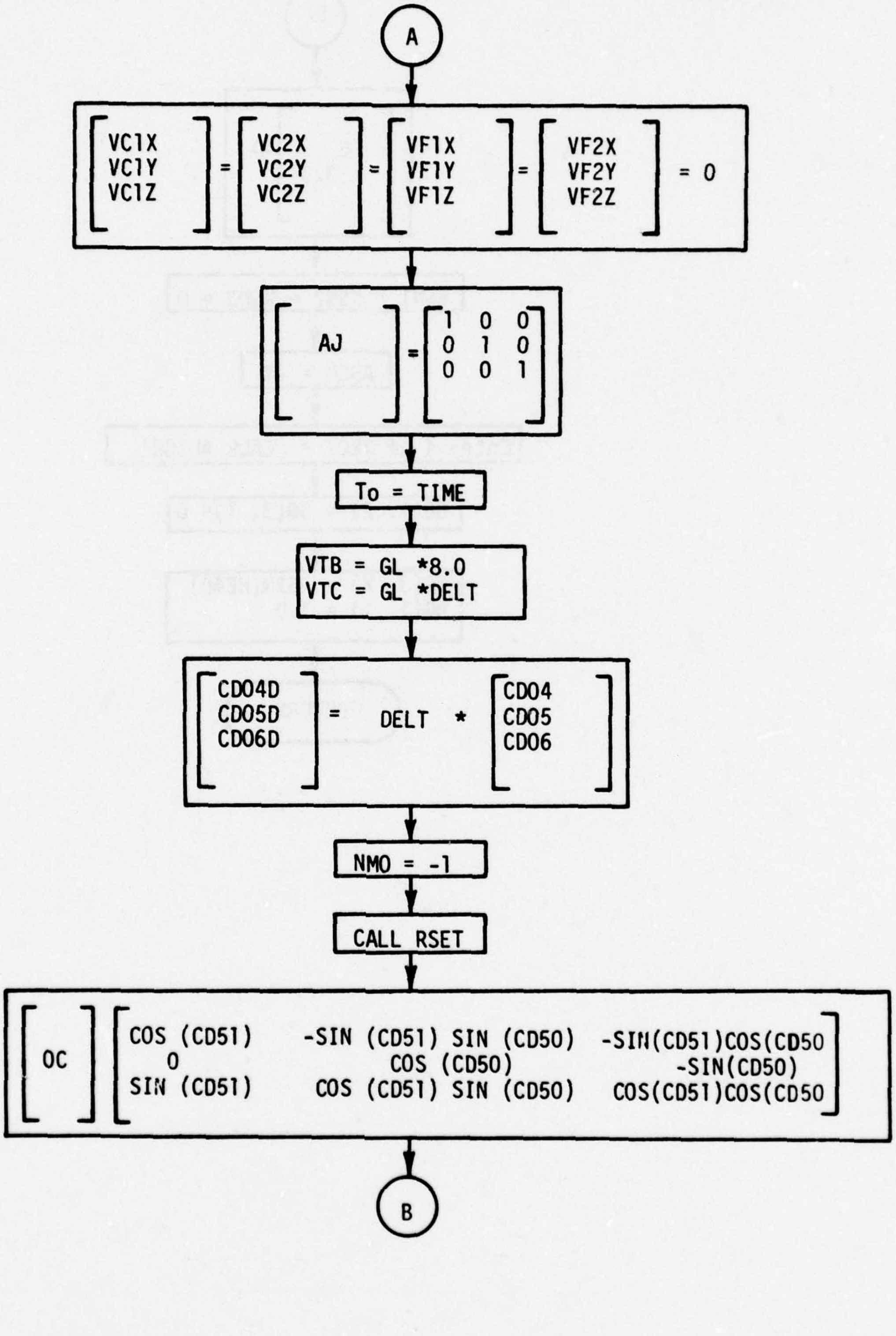
BDSI

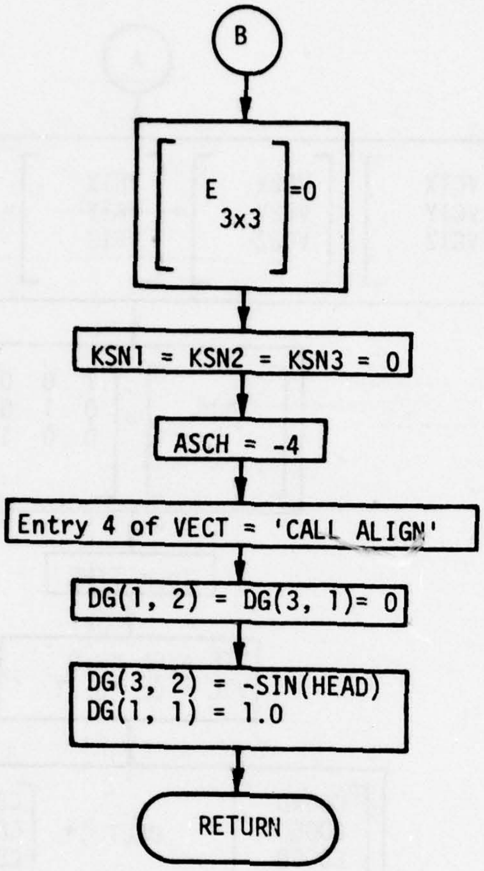

BLP1 = BLP2 = BLP3 = 0
BCTR = BNBR = HCTR = 0
HOLD = MALF = MLFN = 0
CMD1 = CMD2 = CMD4 = 0
BER1 = BER2 = BER3 = BER4=0
BDSI(I) = BTIN(I) I = 1, 4
BMK(I) = BTIN(I+4)I = 1, 4
SRT1 = SRT2 = RATP = RATM = 0
ROT1 = ROT2 = 0
DVXG = DVYG = DVZG = 0
DPVU = DPDV = DPHV = 0
GMT = 0
BTE1 = BTE2 = BTE3 = BTE4 = 0
R1CT = R2CT = CIPM = RAT = RATL = 0
DVX = DVY = DVZ = 0
CYLE = VRTV = DRFV = HDGV = 0
CRT1 = CRT2 = CRT3 = 0
TIME = ITER = PHAS = 0
NAVF = DATA = PUSH = TEST = 0
LITE = KLIT
Ø14 DATA = 0
Ø25 DATA = 0
MODE = MODE SWITCH FROM SIDL
A

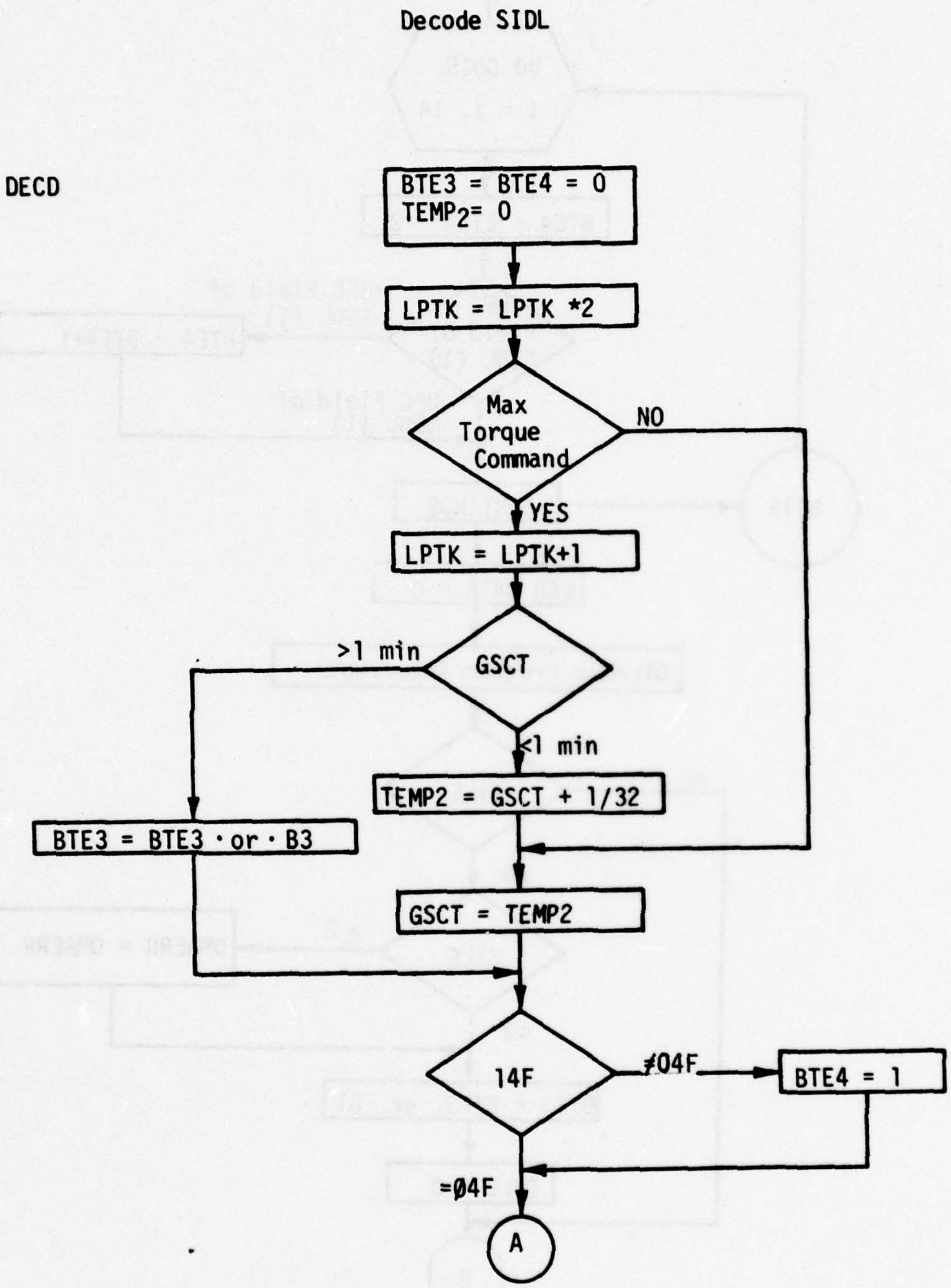

A
2nd Entry of VECT = 'CALL DECD'
5th Entry of VECT = 'CALL SPIN'
7th Entry of VECT = 'CALL BITE'
14th Entry of VECT = GASC
SODL(I) = SODLIN(I) I = 1, 64
RETURN

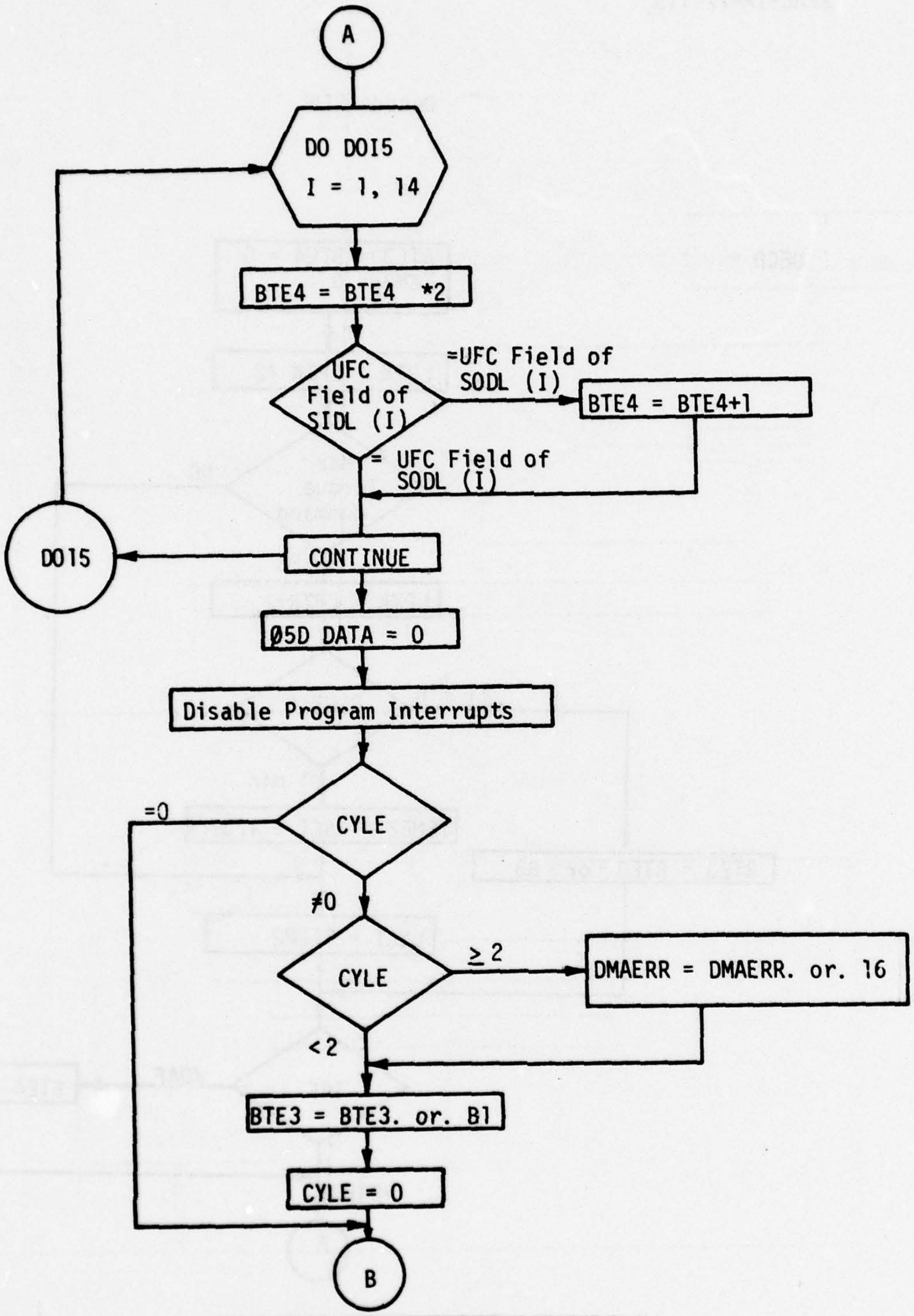
Initialize for Alignment (First Entry)

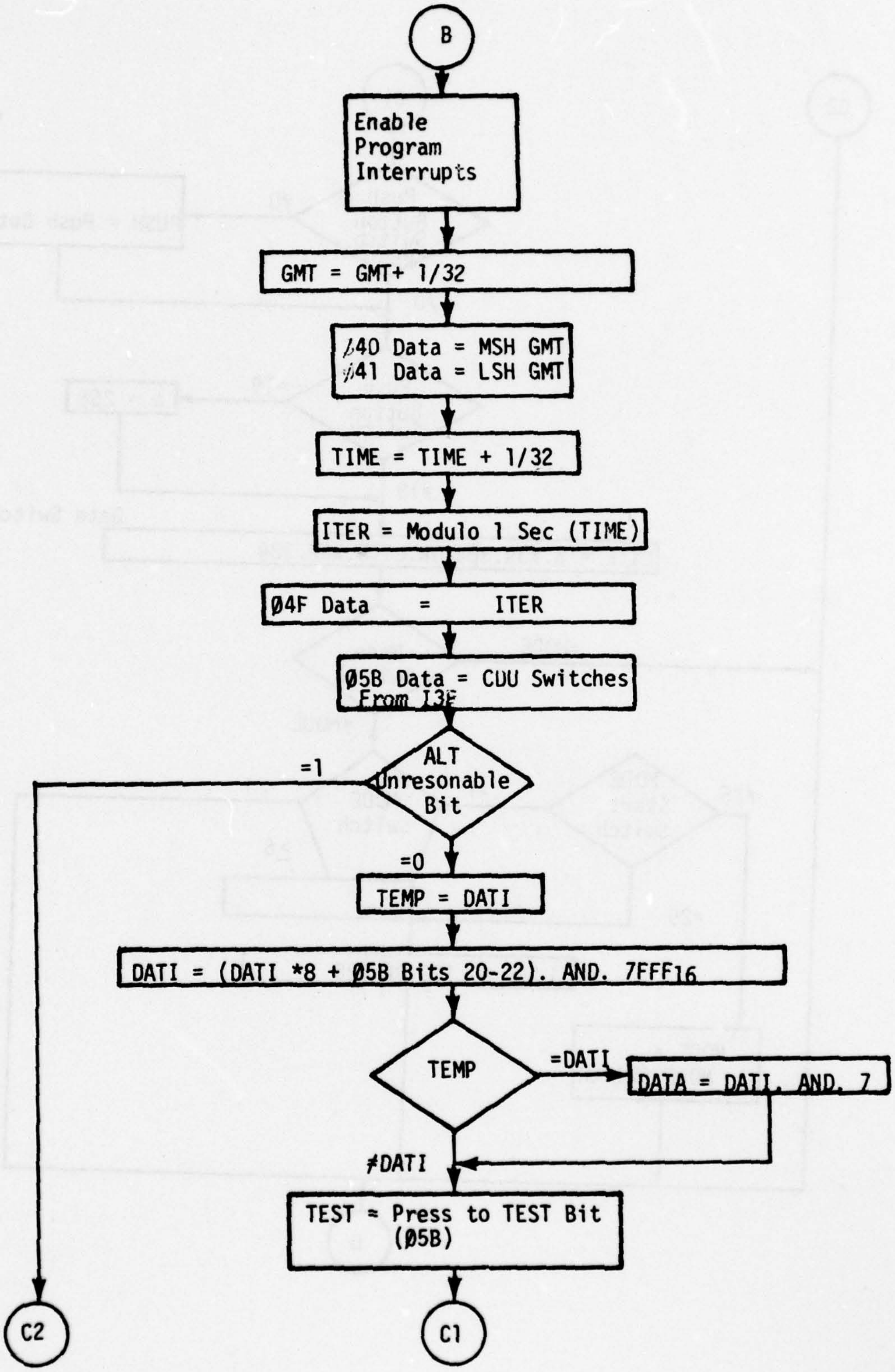

FENT
[A 3x3] = [SDVI SDVJ SDVK] = [VAXI VAYJ VAZK] = [DVXI DVYJ DVZK] = [DVXG DVYG DVZG] = 0
SRTI = SRT2 = RATP = RATM = CHAJ = 0
DCON = -8
[GM 3x3] = -[CD19 CD20 CD21 / CD25 -CD26 -CD27 / CD22 CD23 CD24]
A(1, 1) = CD01
A(2, 2) = CD02
A(3, 3) = CD03
[AB 3x3] = [CD07 CD10 CD13 / CD08 CD11 CD14 / CD09 CD12 CD15] [A 3x3]
LAT = LATL
SGDL = SIN (LAT)
CGDL = COS (LAT)
AK1T = CGDL *GL
AK2T = SGDL *GL
PHA = 0
A

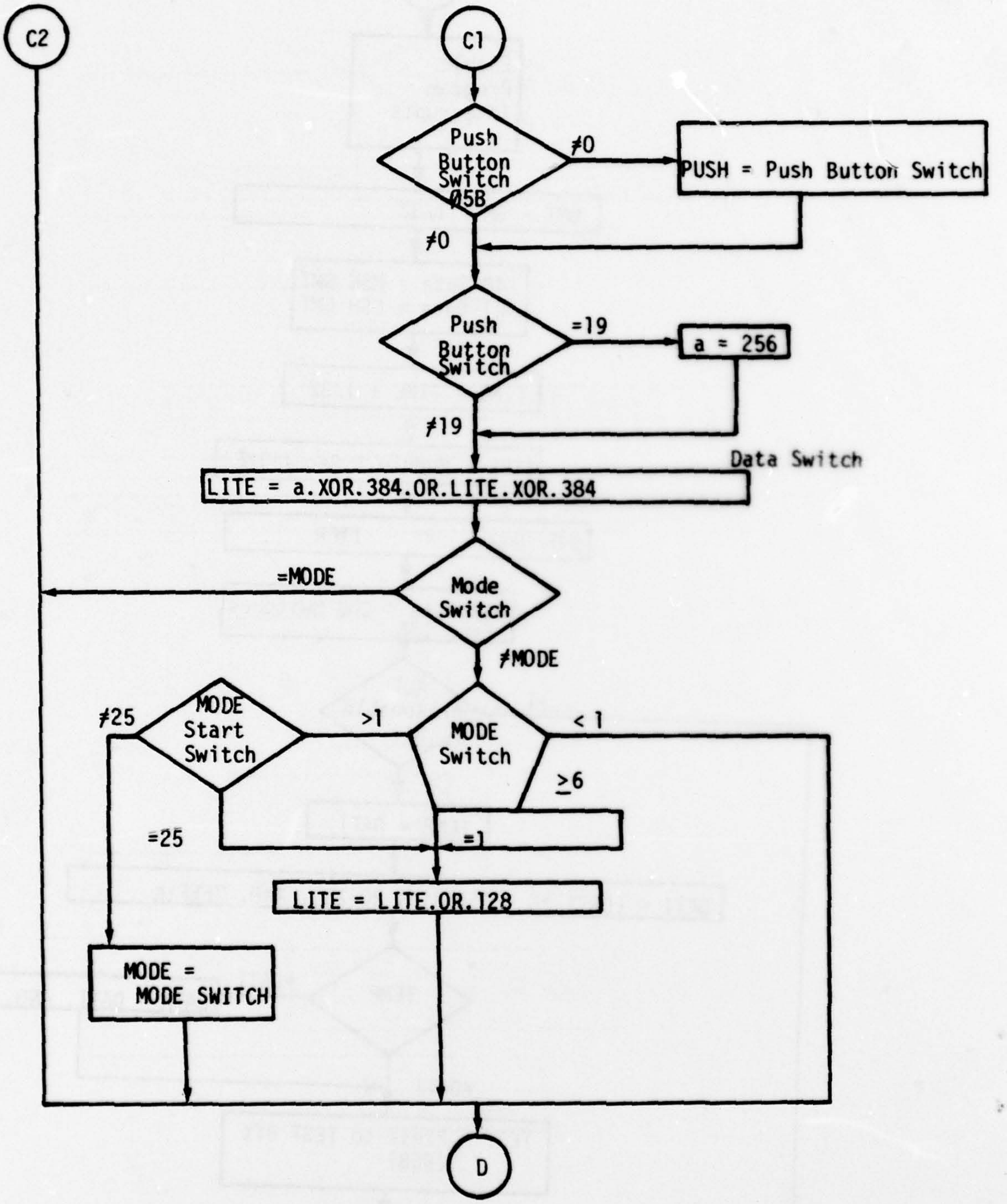

A
[VC1X VC1Y VC1Z] = [VC2X VC2Y VC2Z] = [VF1X VF1Y VF1Z] = [VF2X VF2Y VF2Z] = 0
[AJ] = [1 0 0 / 0 1 0 / 0 0 1]
To = TIME
VTB = GL *8.0
VTC = GL *DELT
[CD04D CD05D CD06D] = DELT * [CD04 CD05 CD06]
NMO = -1
CALL RSET
[OC] [COS (CD51) -SIN (CD51) SIN (CD50) -SIN(CD51)COS(CD50 / 0 COS (CD50) -SIN(CD50) / SIN (CD51) COS (CD51) SIN (CD50) COS(CD51)COS(CD50]
B

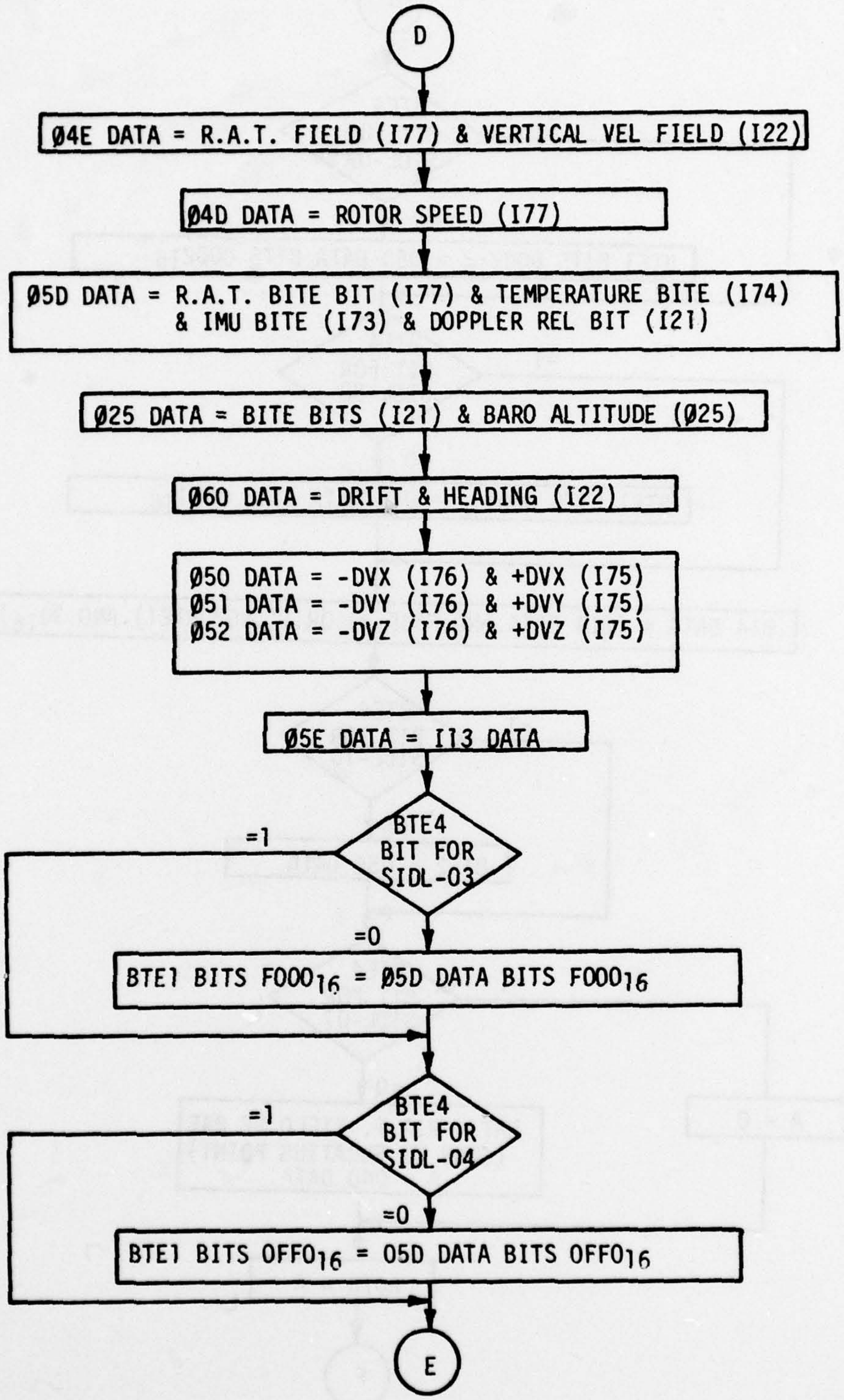

B
[E 3x3] =0
KSN1 = KSN2 = KSN3 = 0
ASCH = -4
Entry 4 of VECT = 'CALL ALIGN'
DG(1, 2) = DG(3, 1)= 0
DG(3, 2) = -SIN(HEAD)
DG(1, 1) = 1.0
RETURN

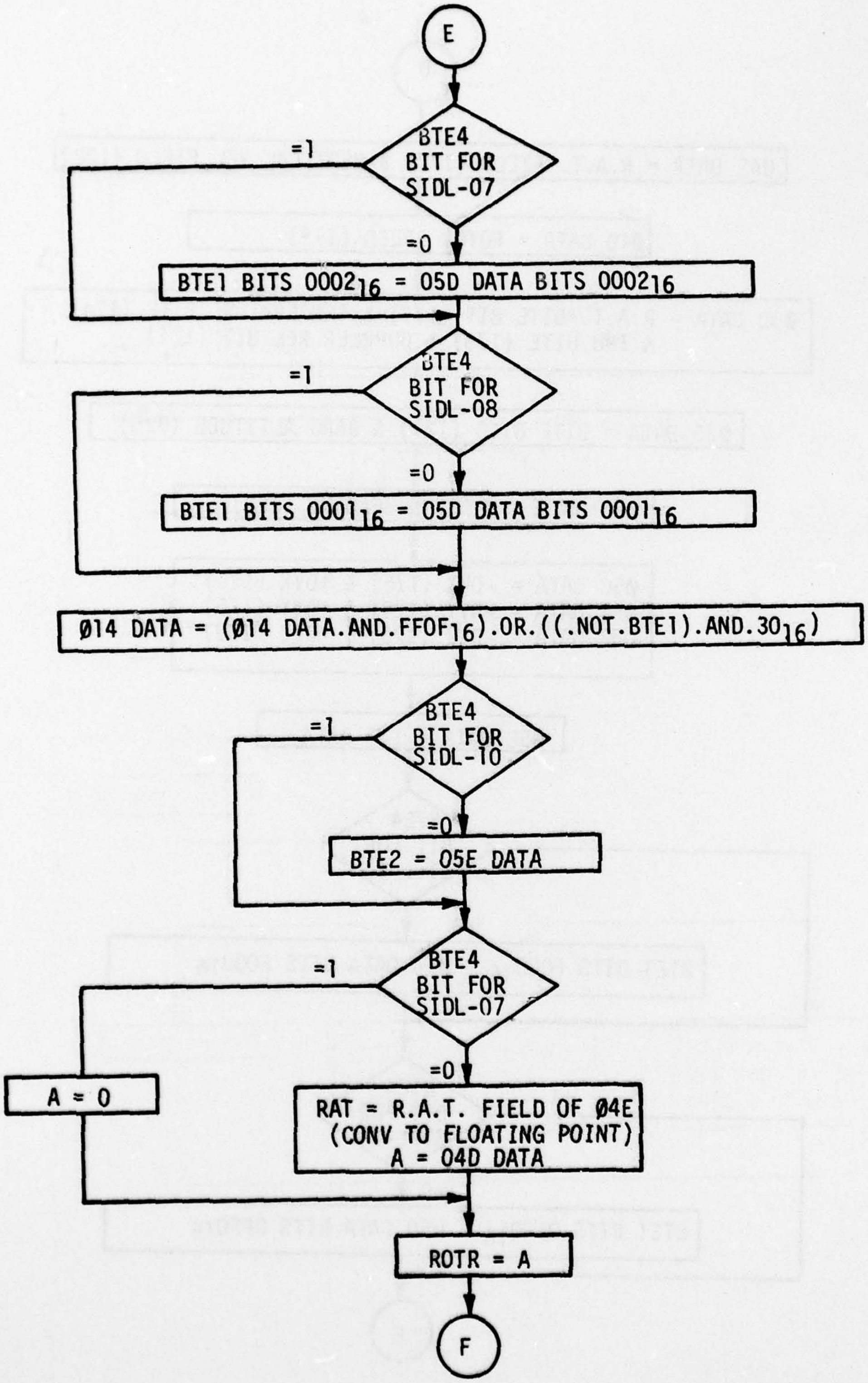

Decode SIDL
DECD
BTE3 = BTE4 = 0
TEMP2= 0
LPTK = LPTK *2
Max Torque Command
NO
YES
LPTK = LPTK+1
GSCT
>1 min
<1 min
TEMP2 = GSCT + 1/32
BTE3 = BTE3 · or · B3
GSCT = TEMP2
14F
≠04F
BTE4 = 1
=Ø4F
A

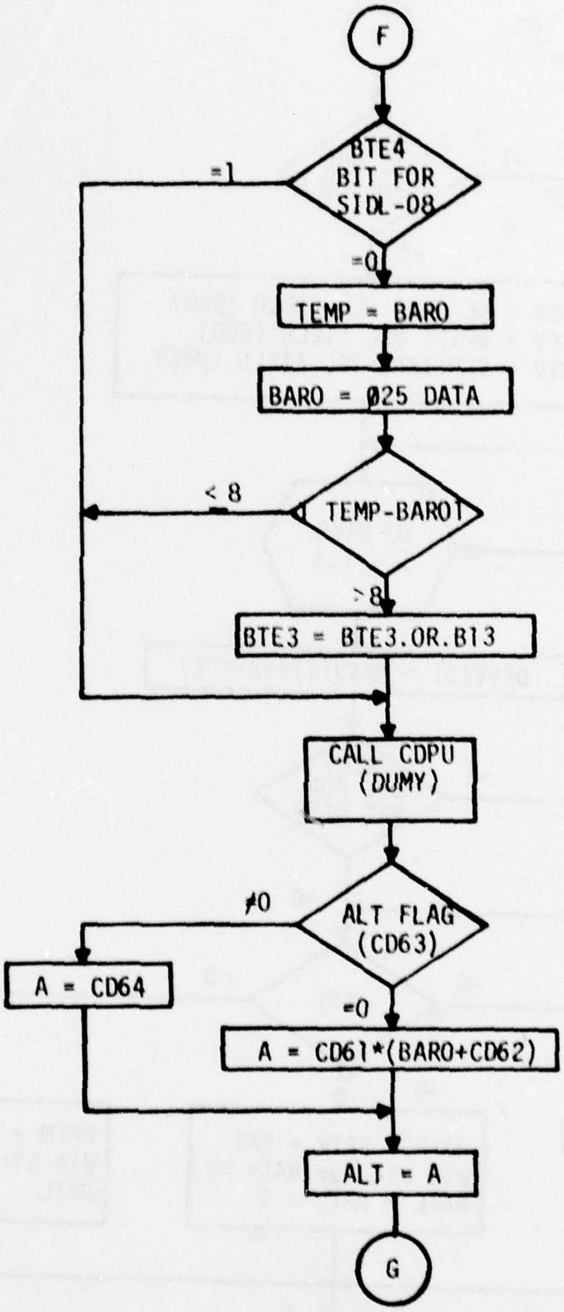

A
DO DOI5
I = 1, 14
BTE4 = BTE4 *2
UFC Field of SIDL (I)
=UFC Field of SODL (I)
BTE4 = BTE4+1
≠ UFC Field of SODL (I)
DO15
CONTINUE
Ø5D DATA = 0
Disable Program Interrupts
CYLE
=0
≠0
CYLE
≥ 2
DMAERR = DMAERR. or. 16
<2
BTE3 = BTE3. or. B1
CYLE = 0
B

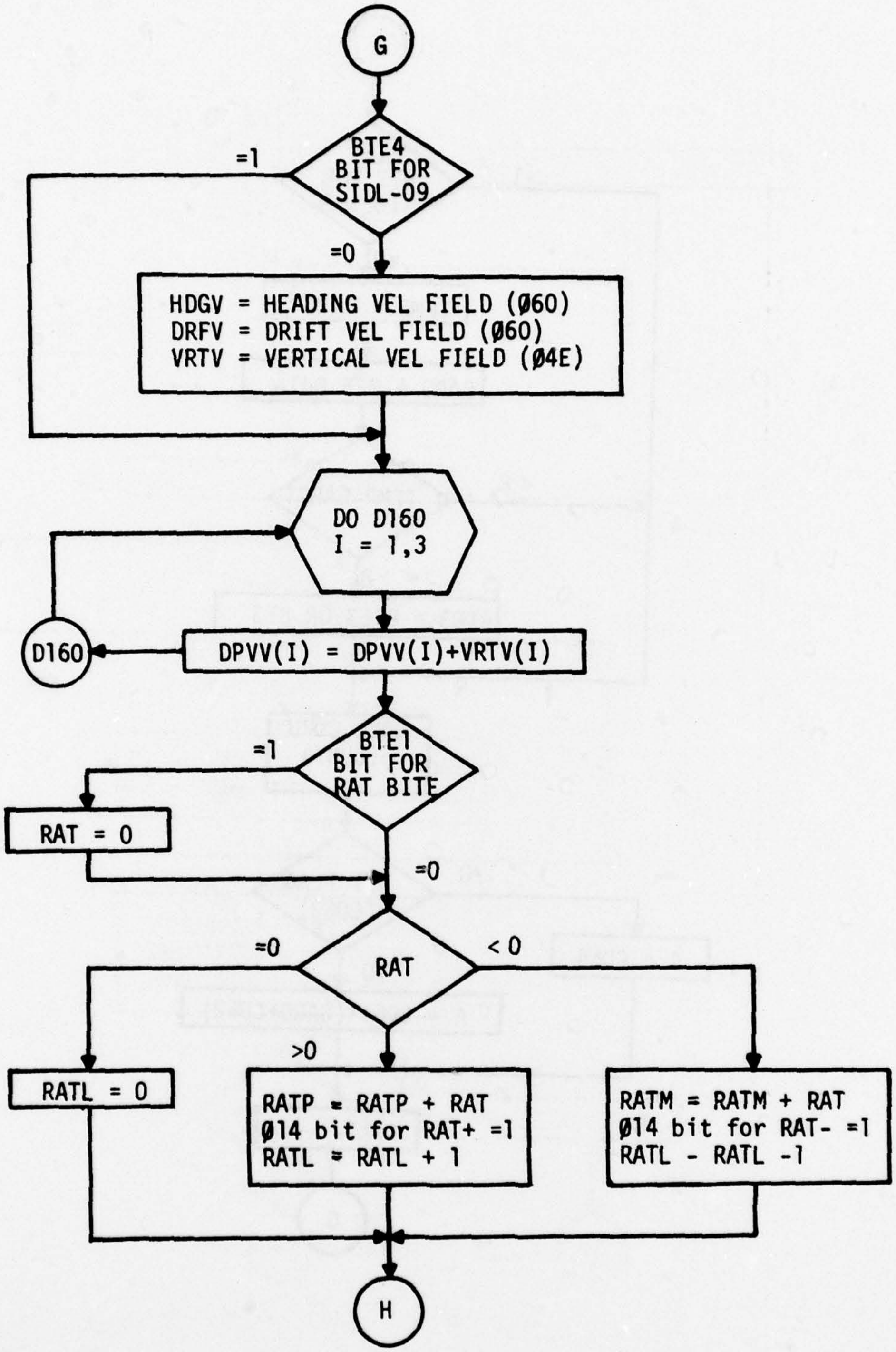

B
Enable Program Interrupts
GMT = GMT+ 1/32
Ø40 Data = MSH GMT
Ø41 Data = LSH GMT
TIME = TIME + 1/32
ITER = Modulo 1 Sec (TIME)
Ø4F Data = ITER
Ø5B Data = CDU Switches From I3E
=1
ALT Unresonable Bit
=0
TEMP = DATI
DATI = (DATI *8 + Ø5B Bits 20-22). AND. 7FFF16
TEMP
=DATI
DATA = DATI. AND. 7
≠DATI
TEST = Press to TEST Bit (Ø5B)
C2
C1

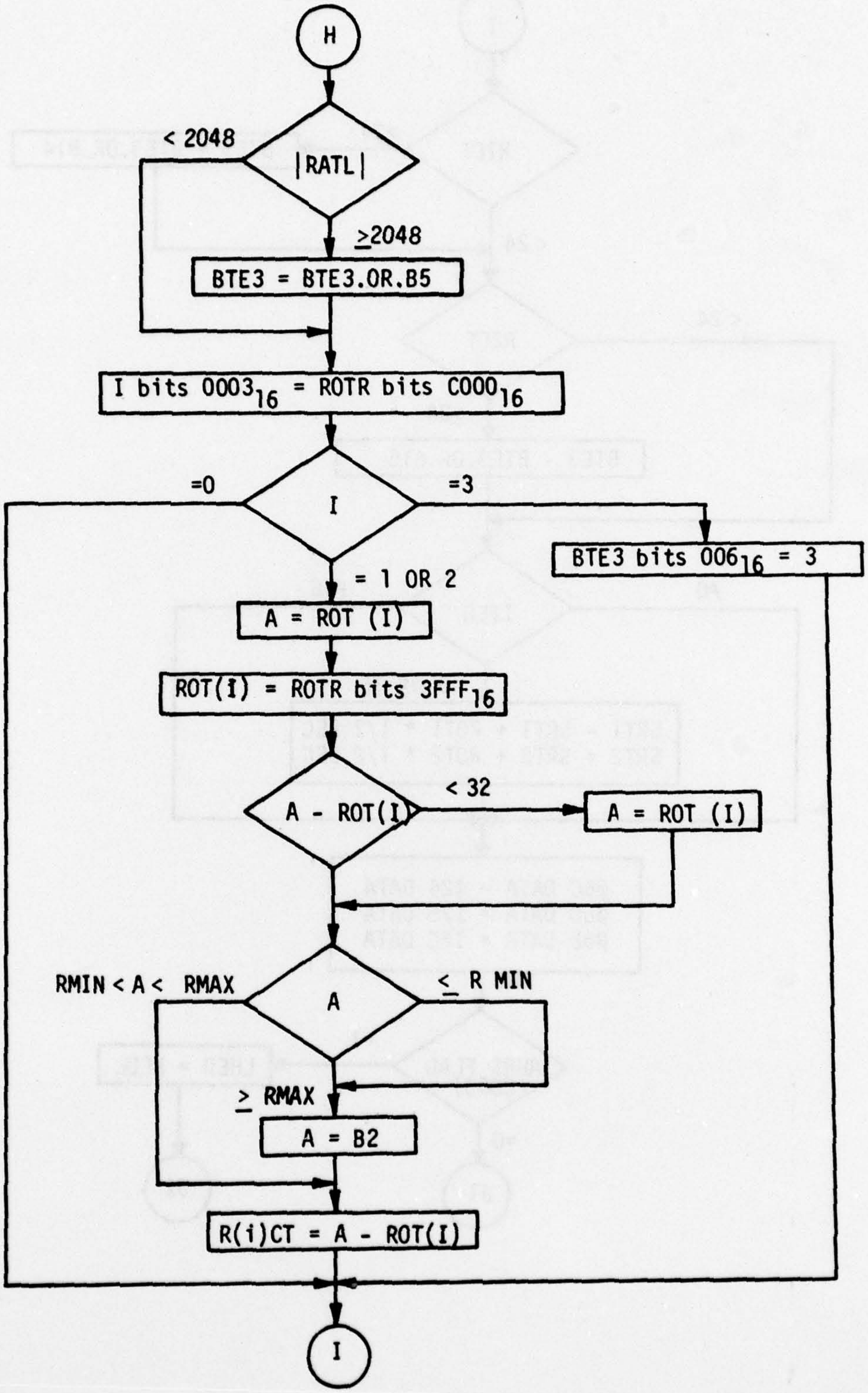

C2
C1
Push Button Switch Ø5B
≠0
PUSH = Push Button Switch
≠0
Push Button Switch
=19
a = 256
≠19
Data Switch
LITE = a.XOR.384.OR.LITE.XOR.384
=MODE
Mode Switch
≠MODE
≠25
MODE Start Switch
>1
MODE Switch
< 1
≥6
=25
=1
LITE = LITE.OR.128
MODE = MODE SWITCH
D

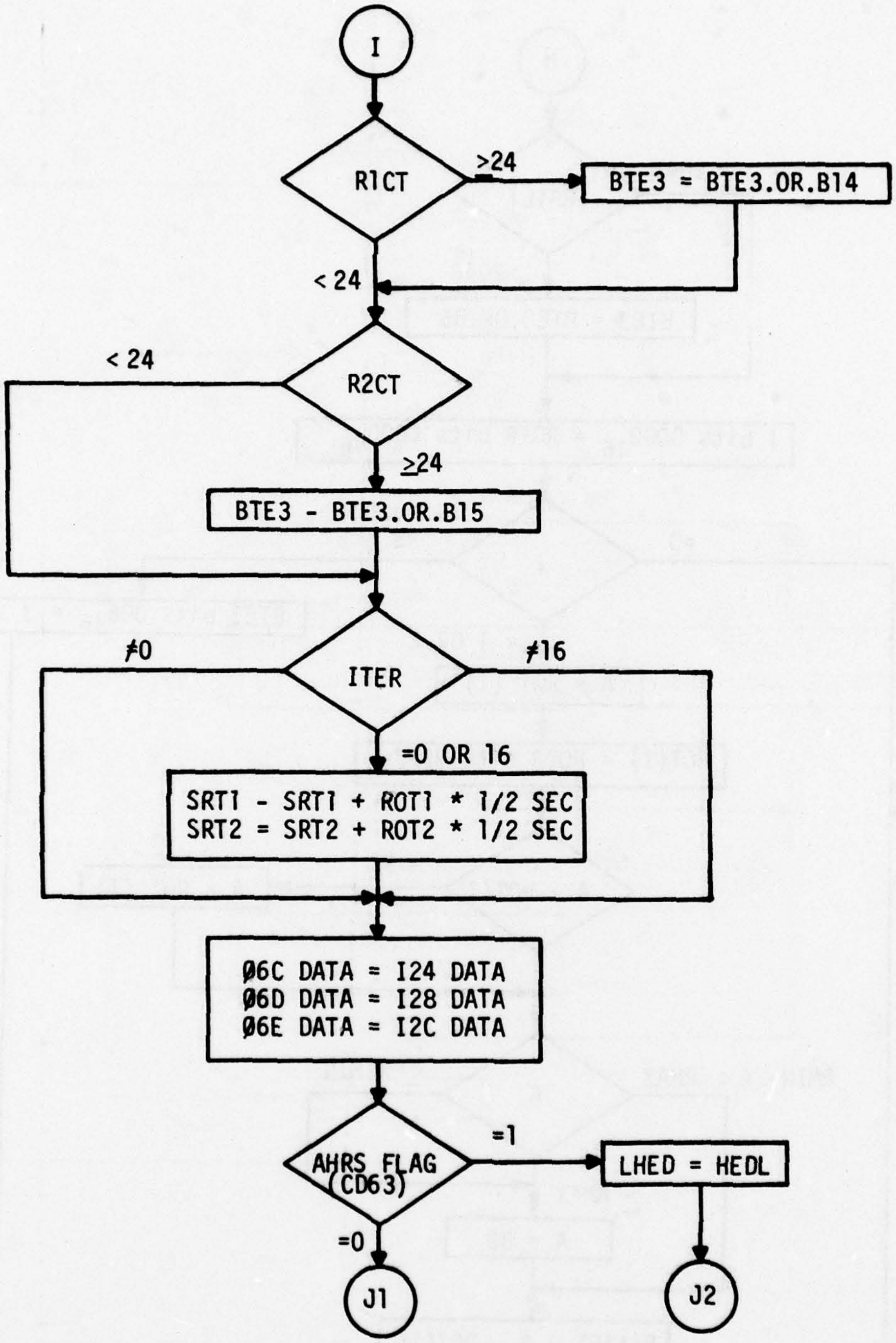

D
Ø4E DATA = R.A.T. FIELD (I77) & VERTICAL VEL FIELD (I22)
Ø4D DATA = ROTOR SPEED (I77)
Ø5D DATA = R.A.T. BITE BIT (I77) & TEMPERATURE BITE (I74) & IMU BITE (I73) & DOPPLER REL BIT (I21)
Ø25 DATA = BITE BITS (I21) & BARO ALTITUDE (Ø25)
Ø60 DATA = DRIFT & HEADING (I22)
Ø50 DATA = -DVX (I76) & +DVX (I75)
Ø51 DATA = -DVY (I76) & +DVY (I75)
Ø52 DATA = -DVZ (I76) & +DVZ (I75)
Ø5E DATA = I13 DATA
BTE4 BIT FOR SIDL-03
=1
=0
BTE1 BITS F000$_{16}$ = Ø5D DATA BITS F000$_{16}$
BTE4 BIT FOR SIDL-04
=1
=0
BTE1 BITS OFFO$_{16}$ = 05D DATA BITS OFFO$_{16}$
E

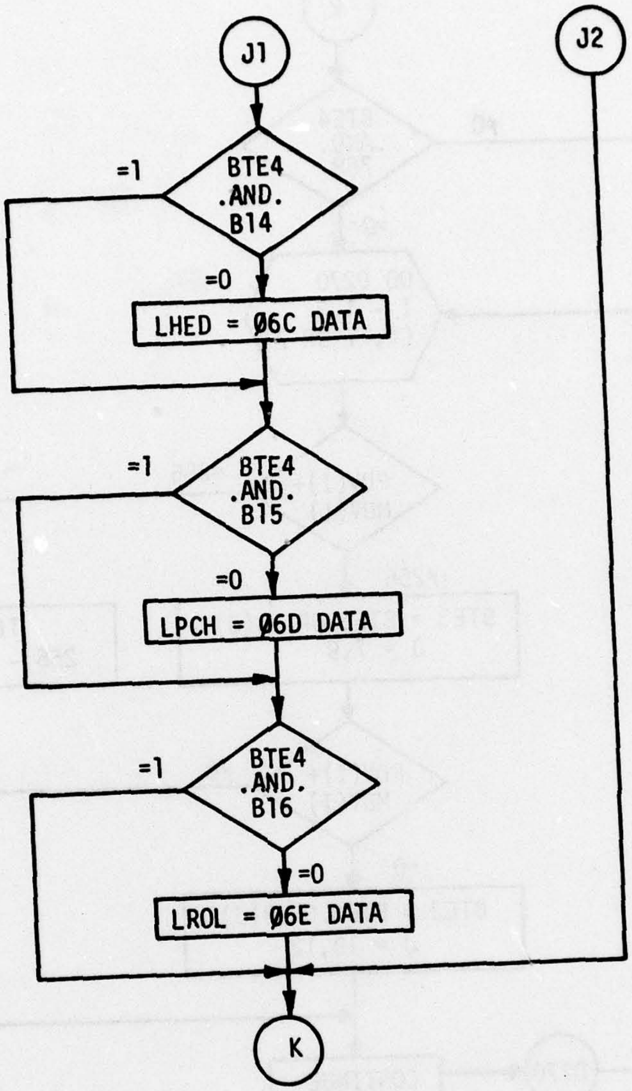

E
BTE4 BIT FOR SIDL-07
=1
=0
BTE1 BITS 0002₁₆ = 05D DATA BITS 0002₁₆
BTE4 BIT FOR SIDL-08
=1
=0
BTE1 BITS 0001₁₆ = 05D DATA BITS 0001₁₆
Ø14 DATA = (Ø14 DATA.AND.FFOF₁₆).OR.((.NOT.BTE1).AND.30₁₆)
BTE4 BIT FOR SIDL-10
=1
=0
BTE2 = 05E DATA
BTE4 BIT FOR SIDL-07
=1
=0
A = 0
RAT = R.A.T. FIELD OF Ø4E (CONV TO FLOATING POINT) A = 04D DATA
ROTR = A
F

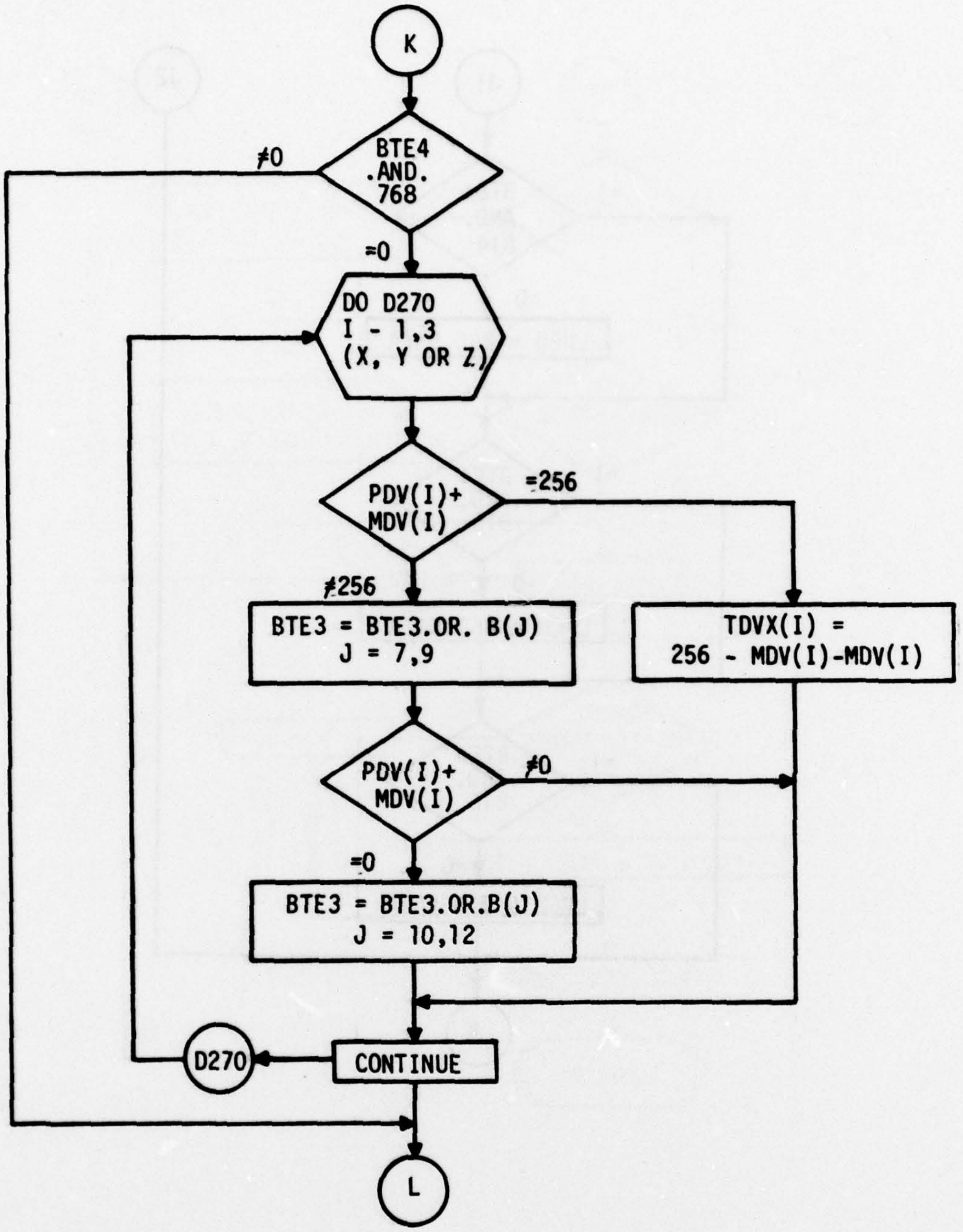

F
BTE4
BIT FOR
SIDL-08
=1
=0
TEMP = BARO
BARO = Ø25 DATA
TEMP-BARO1
< 8
> 8
BTE3 = BTE3.OR.B13
CALL CDPU
(DUMY)
ALT FLAG
(CD63)
≠0
=0
A = CD64
A = CD61*(BARO+CD62)
ALT = A
G

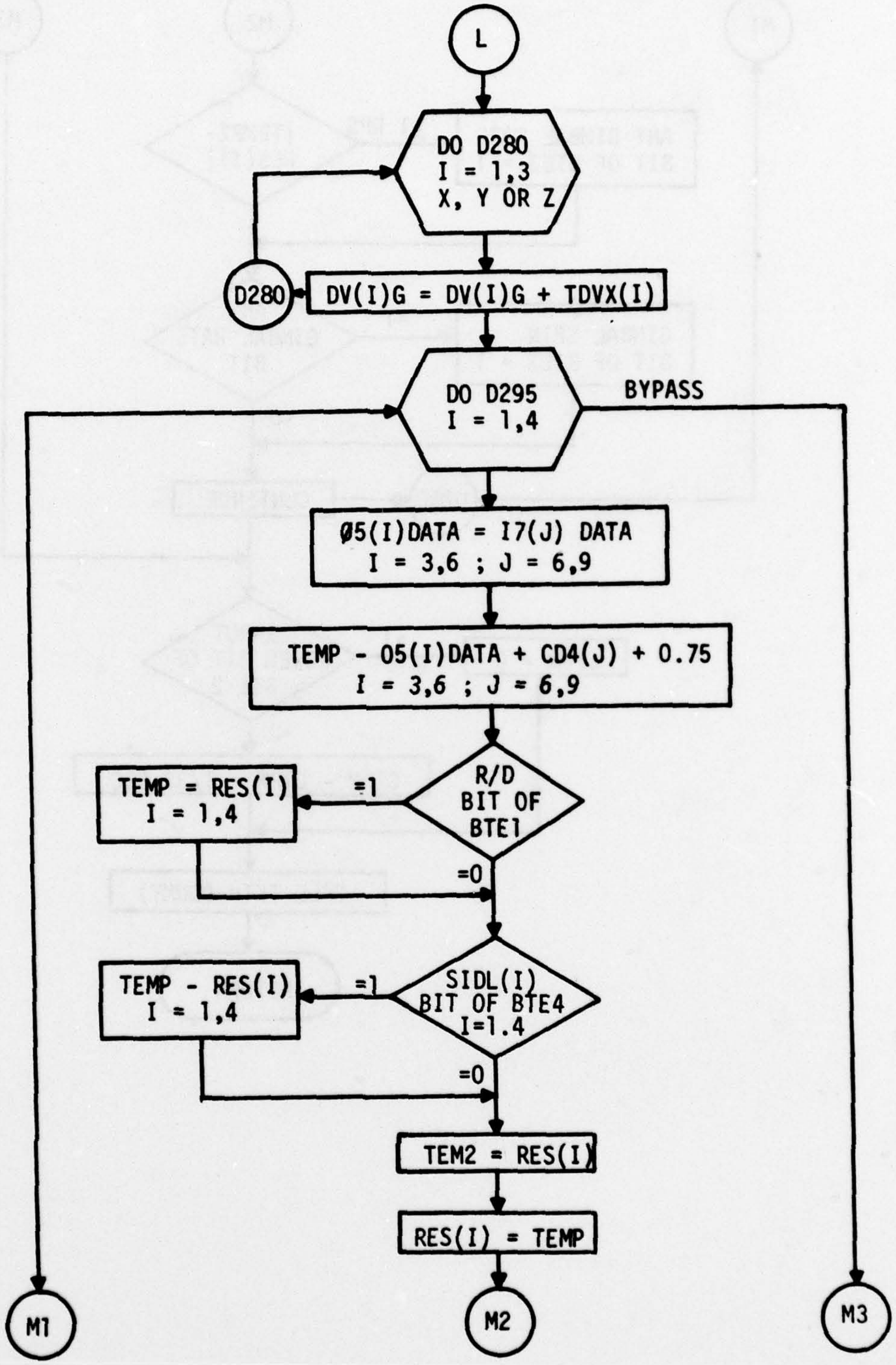

G
BTE4
BIT FOR
SIDL-09
=1
=0
HDGV = HEADING VEL FIELD (Ø60)
DRFV = DRIFT VEL FIELD (Ø60)
VRTV = VERTICAL VEL FIELD (Ø4E)
DO D160
I = 1,3
D160
DPVV(I) = DPVV(I)+VRTV(I)
BTE1
BIT FOR
RAT BITE
=1
RAT = 0
=0
RAT
=0
< 0
>0
RATL = 0
RATP = RATP + RAT
Ø14 bit for RAT+ =1
RATL = RATL + 1
RATM = RATM + RAT
Ø14 bit for RAT- =1
RATL - RATL -1
H

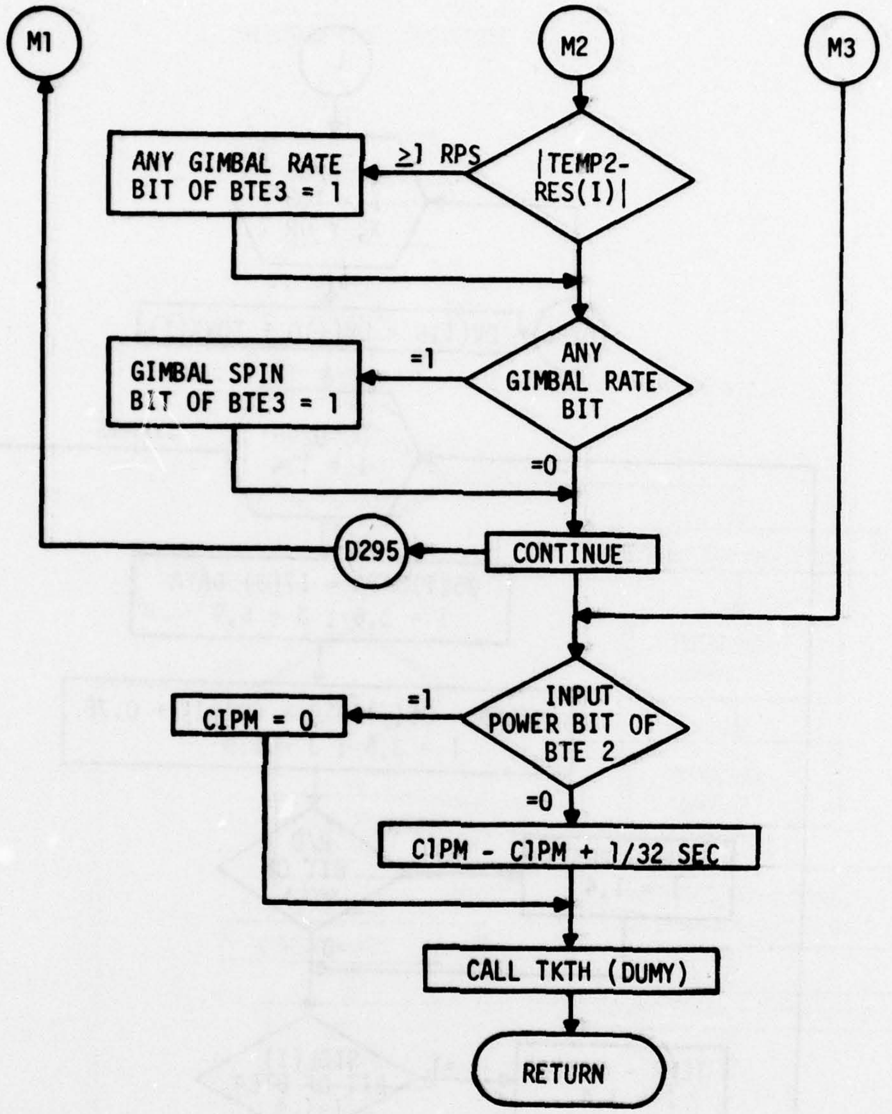

H
|RATL|
< 2048
≥2048
BTE3 = BTE3.OR.B5
I bits 0003₁₆ = ROTR bits C000₁₆
I
=0
=3
= 1 OR 2
BTE3 bits 006₁₆ = 3
A = ROT (I)
ROT(I) = ROTR bits 3FFF₁₆
A - ROT(I)
< 32
A = ROT (I)
A
RMIN < A < RMAX
≤ R MIN
≥ RMAX
A = B2
R(i)CT = A - ROT(I)
I


I
R1CT
>24
BTE3 = BTE3.OR.B14
< 24
R2CT
< 24
>24
BTE3 - BTE3.OR.B15
ITER
≠0
≠16
=0 OR 16
SRT1 - SRT1 + ROT1 * 1/2 SEC
SRT2 = SRT2 + ROT2 * 1/2 SEC
Ø6C DATA = I24 DATA
Ø6D DATA = I28 DATA
Ø6E DATA = I2C DATA
AHRS FLAG (CD63)
=1
LHED = HEDL
=0
J1
J2

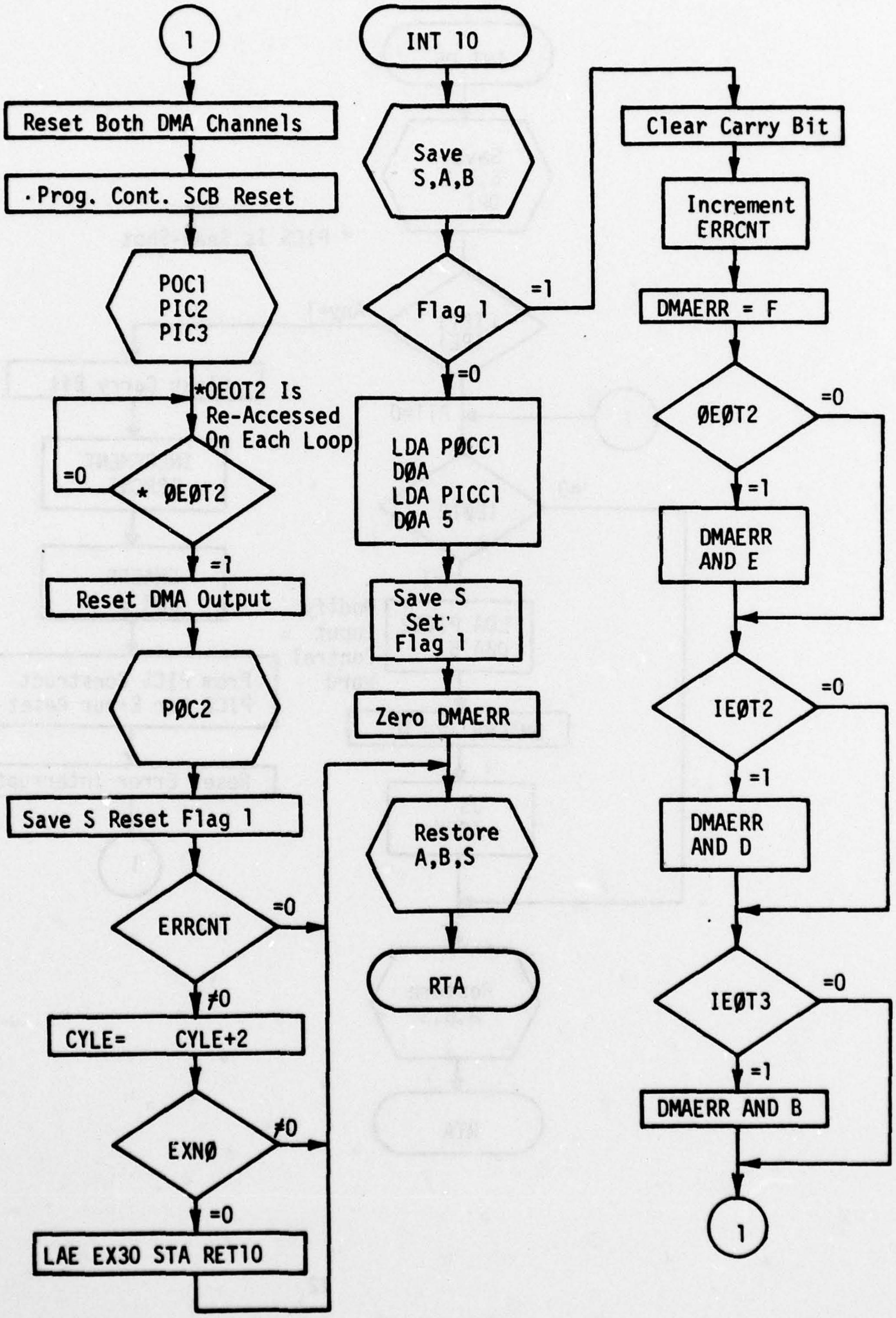

J1
J2
BTE4
.AND.
B14
=1
=0
LHED = Ø6C DATA
BTE4
.AND.
B15
=1
=0
LPCH = Ø6D DATA
BTE4
.AND.
B16
=1
=0
LROL = Ø6E DATA
K


K
BTE4
.AND.
768
≠0
=0
DO D270
I - 1,3
(X, Y OR Z)
PDV(I)+
MDV(I)
=256
≠256
BTE3 = BTE3.OR. B(J)
J = 7,9
TDVX(I) =
256 - MDV(I)-MDV(I)
PDV(I)+
MDV(I)
≠0
=0
BTE3 = BTE3.OR.B(J)
J = 10,12
CONTINUE
D270
L

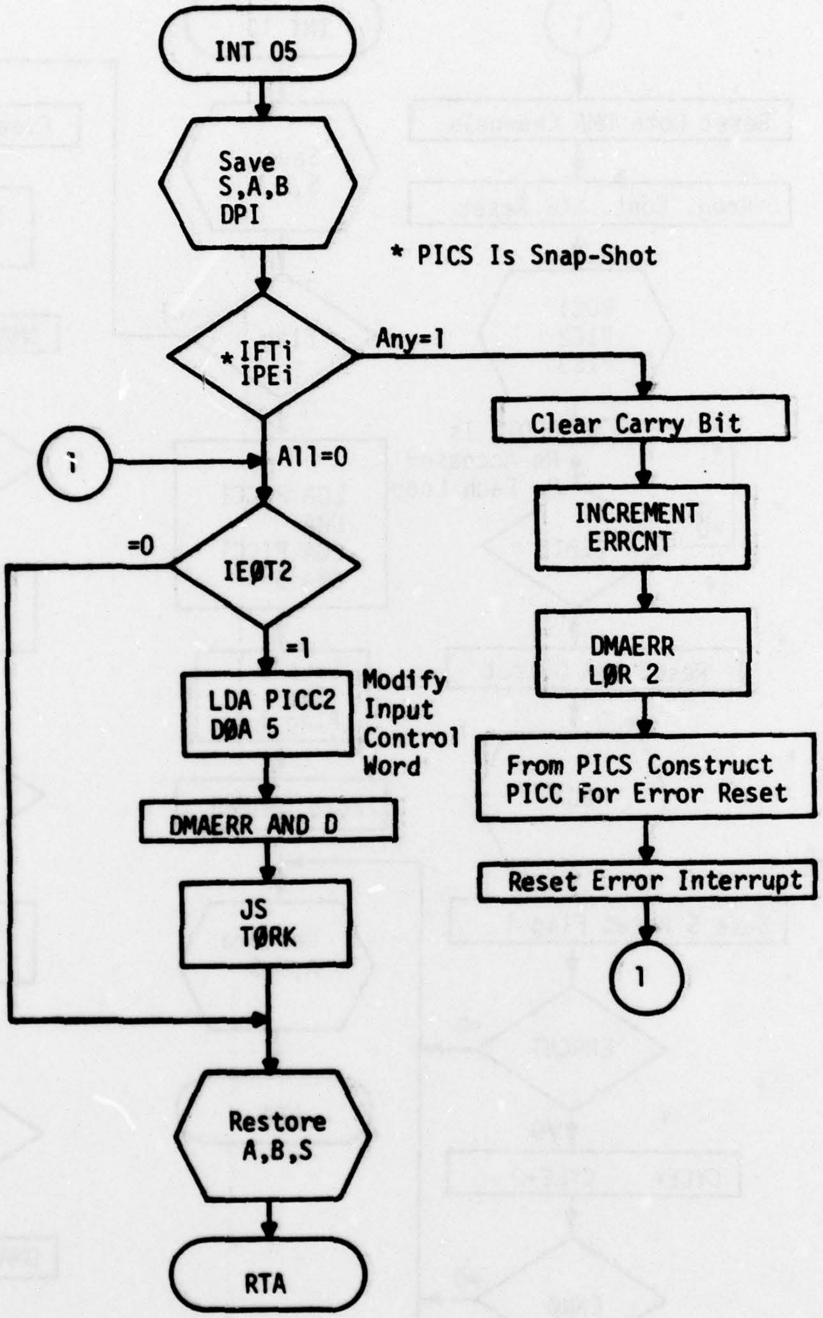

L
DO D280
I = 1,3
X, Y OR Z
D280
DV(I)G = DV(I)G + TDVX(I)
DO D295
I = 1,4
BYPASS
Ø5(I)DATA = I7(J) DATA
I = 3,6 ; J = 6,9
TEMP - O5(I)DATA + CD4(J) + 0.75
I = 3,6 ; J = 6,9
TEMP = RES(I)
I = 1,4
=1
R/D
BIT OF
BTE1
=0
TEMP - RES(I)
I = 1,4
=1
SIDL(I)
BIT OF BTE4
I=1.4
=0
TEM2 = RES(I)
RES(I) = TEMP
M1
M2
M3


M1
M2
M3
ANY GIMBAL RATE BIT OF BTE3 = 1
≥1 RPS
|TEMP2-RES(I)|
GIMBAL SPIN BIT OF BTE3 = 1
=1
ANY GIMBAL RATE BIT
=0
D295
CONTINUE
CIPM = 0
=1
INPUT POWER BIT OF BTE 2
=0
CIPM - CIPM + 1/32 SEC
CALL TKTH (DUMY)
RETURN

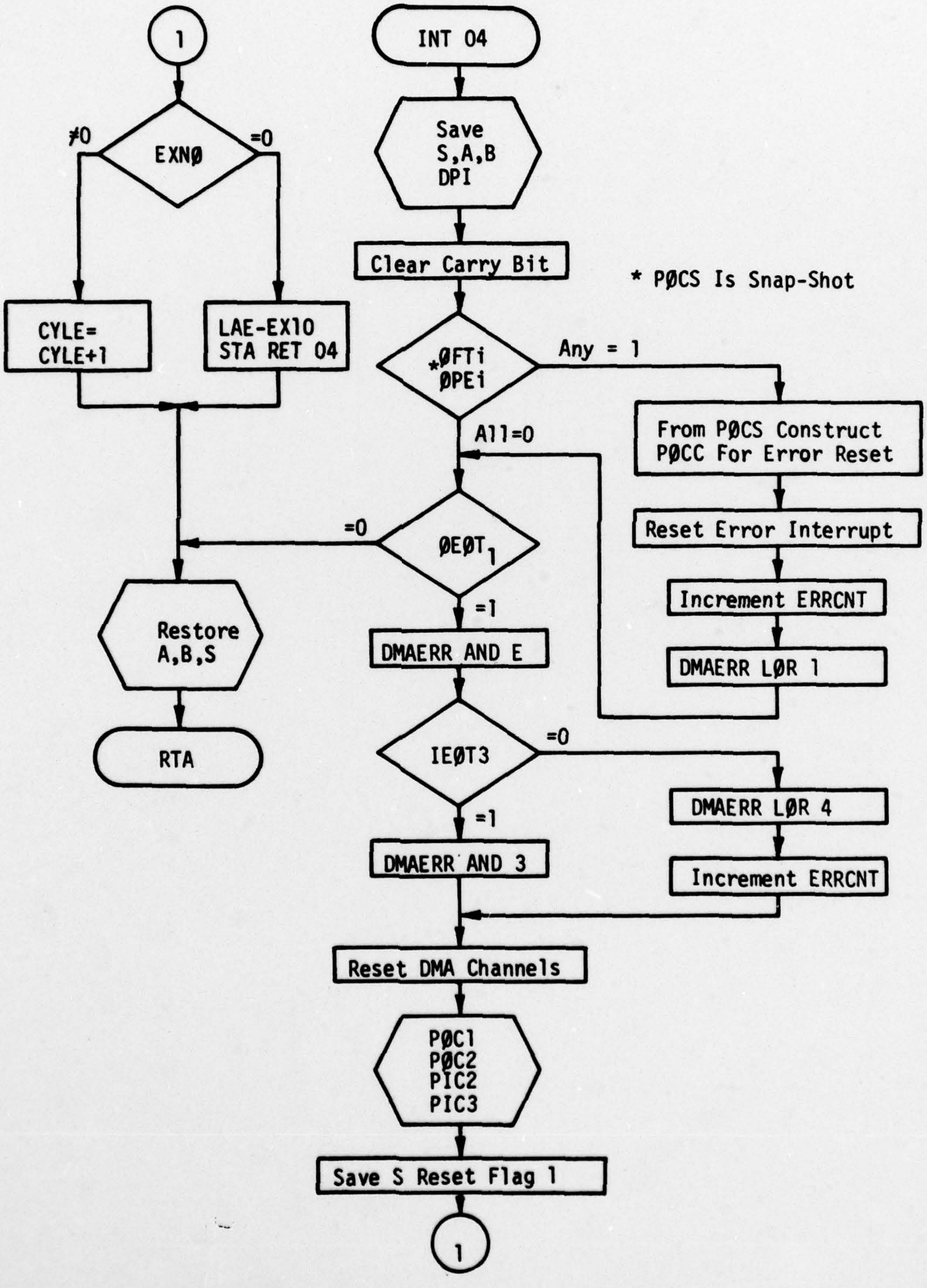
INTERRUPT 10 ROUTINE [32 HZ]



INTERRUPT 5 ROUTINE



INTERRUPT 4 ROUTINE


1
EXNØ
≠0
=0
CYLE=
CYLE+1
LAE-EX10
STA RET 04
Restore
A,B,S
RTA
INT 04
Save
S,A,B
DPI
Clear Carry Bit
* PØCS Is Snap-Shot
*ØFTi
ØPEi
Any = 1
All=0
From PØCS Construct
PØCC For Error Reset
Reset Error Interrupt
Increment ERRCNT
DMAERR LØR 1
ØEØT1
=0
=1
DMAERR AND E
IEØT3
=0
=1
DMAERR LØR 4
Increment ERRCNT
DMAERR AND 3
Reset DMA Channels
PØC1
PØC2
PIC2
PIC3
Save S Reset Flag 1
1