2
mix

AEROSPACE REPORT NO.
ATR-73(7315)-1. VOL. II

NTIS HC $14.50

DORCA II Computer Program
Volume II: Programmer's Manual

(NASA-CR-129264) DORCA 2 COMPUTER PROGRAM. VOLUME 2: PROGRAMMER'S MANUAL B.J. Gold (Aerospace Corp., El Segundo, Calif.) 18 Aug. 1972 247 p CSCL 09B

N73-12184

Unclas
G3/08 48325

Prepared by BARBARA J. GOLD
Information Processing Division

18 August 1972

Prepared for OFFICE OF MANNED SPACE FLIGHT
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Washington, D. C.

Contract No. NASw-2301

Systems Engineering Operations
THE AEROSPACE CORPORATION

Report No. ATR-73(7315)-1, Vol. II

DORCA II COMPUTER PROGRAM

Volume II: Programmer's Manual

Prepared by

Barbara J. Gold
Vehicle Analysis Programming Department
Information Processing Division
Engineering Science Operations

18 August 1972

Systems Engineering Operations
THE AEROSPACE CORPORATION
El Segundo, California

Prepared for

Office of Manned Space Flight
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Washington, D. C.

Contract No. NASw 2301

Report No.
ATR-73(7315)-1, Vol. II

DORCA II COMPUTER PROGRAM:
Volume II: Programmer's Manual

Prepared by

Barbara J. Gold

Barbara J. Gold
Evaluation & Planning Section
Vehicle Analysis Programming Dept.

Approved by

G. W. Timpson

G. W. Timpson, Head
Vehicle Analysis Programming Dept.
Mathematics & Programming
Subdivision

Robert R. Wolfe

Robert R. Wolfe, Director
Operations Office
Advanced Vehicle Systems
Directorate
Systems Planning Division

ABSTRACT

This manual is provided for the use of individuals studying the coding of program DORCA. It explains the detailed operation of every subroutine, the layout in core of the major matrices and arrays, and the meaning of all program variables. Flow charts are included. This volume does not contain user information such as input requirements; such information is found in volume I.

TABLE OF CONTENTS

1	General Program Organization	1
2	Subroutine ASINER	5
3	Subroutine CNTRPT	31
4	Subroutine COLECT	34
5	Subroutine CSTRPT	35
6	Subroutine DDBSFT	42
7.	Routine DORCA	44
8	Subroutine DPAGER	46
9	Subroutine FACNUM	49
10	Subroutine FACRPT	52
11	Subroutine FIND	54
12	Subroutine FINDSP	60
13	Subroutine INPRO	62
14	Subroutine LEGPRO	64
15	Subroutine L2WO	70
16	Subroutine MERGE	71
17	Subroutine MOREL2	79
18	Subroutine PACK	81
19	Subroutine PERLNK	84
20	Subroutine PROLNK	86
21	Subroutine PROPCL	88
22	Subroutine RDCONT	92
23	Subroutine RDCRG	95
24	Subroutine RDFAC	99
25	Subroutine RDLEG	101
26	Subroutine RDMISS	104
27	Subroutine RDRPT	111
28	Subroutine RDSPD	114
29	Subroutine RDVEH	118

30	Subroutine READER	125
31	Subroutine REPORT	127
32	Subroutine SEG22, SEG32, SEG33, SEG36	129
33	Subroutine SORT	130
34	Subroutine SPDAP	134
35	Subroutine SPRINT	138
36	Subroutine SWITCH	142
37	Subroutine TABLES	143
38	Subroutine TRAFIC	145
39	Subroutine UNPACK	155
40	Subroutine VALUE	156
41	Subroutine VEHLDF	159
42	Subroutine VEHRPT	162
43	Subroutine YEARS	165
APPENDIX A:	MASTER NOMENCLATURE LIST	A-1
APPENDIX B:	MAJOR TABLES AND ARRAYS	B-1

AN OVERVIEW OF AN OVERLAY STRUCTURE

OVERLAY 0-0

DORCA
SORT
PACK
MXIFT

- INPRO
- LEGPRO
- REPORT

OVERLAY 1-0

INPRO
SWITCH
READER
VALUE
FINDSP

- RDCONT
- RDLEG
- RDSPD
- RDVEH, PERLNK, PROPCL*
- RDFAC
- RDCRG
- RDMISS, DDBSFT**
- RDRPT

OVERLAY 2-0

LEGPRO
COLECT
L2WO
YEARS
DDBSFT**
PROLNK
PROPCL*

- ASINER, FIND
- SEG22, TRAFIC***

OVERLAY 3-0

REPORT
MOREL2
FACNUM
VEHLDF

- TABLES
- SEG22, MERGE
- SEG33, SPRINT
- CNTRPT
- FACRPT
- SEG36, TRAFIC***
- VEHRPT
- CSTRPT, SPDAP, DPAGER

*, **, *** - SUBROUTINES THAT APPEAR TWICE

SECTION 1

GENERAL PROGRAM ORGANIZATION

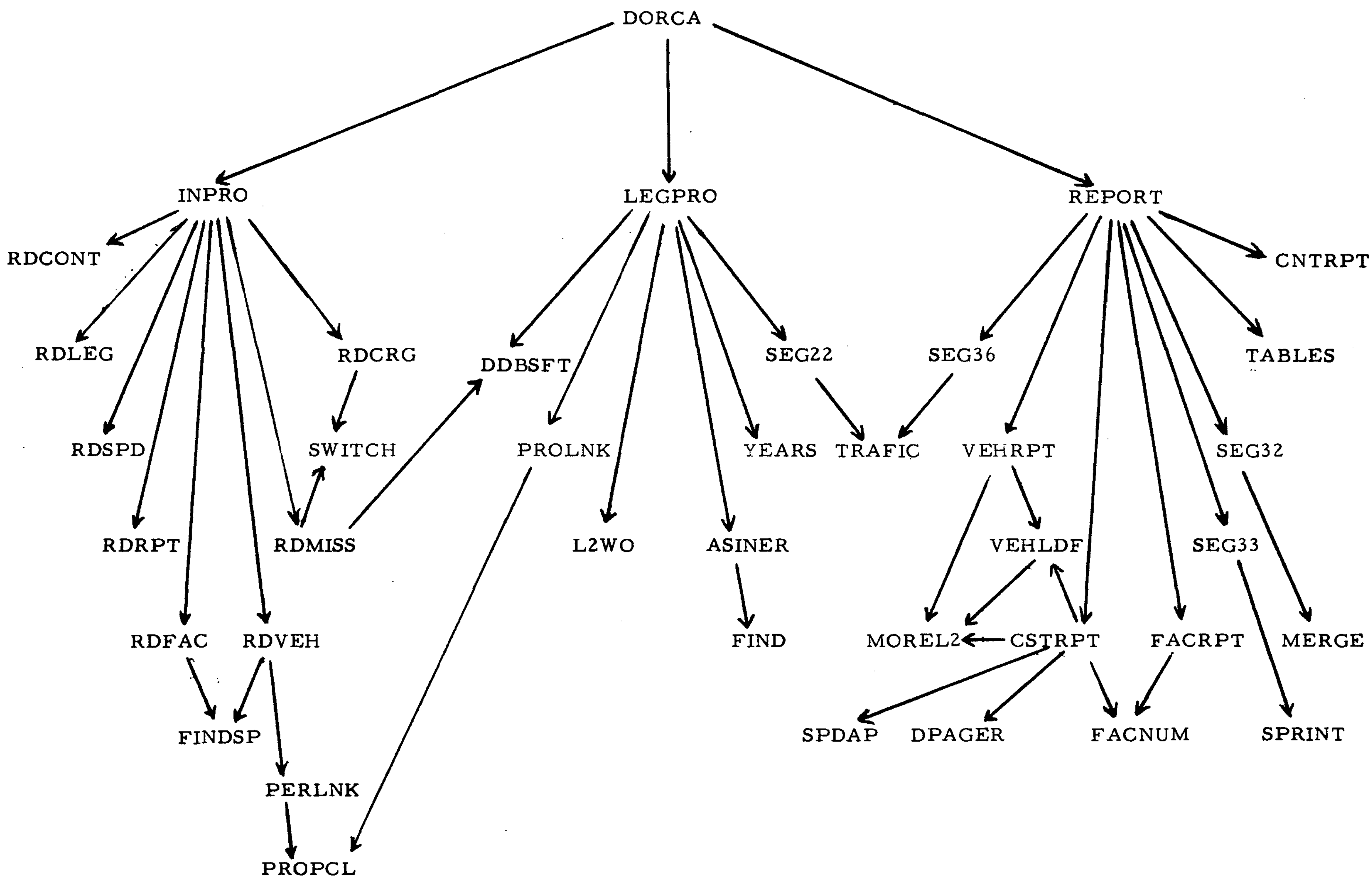
DORCA is logically divided into three major sections: input, computation, and output. Each of these sections is treated extensively in the user's manual in a problem-oriented manner. The following notes are a general outline of the program mechanics.

Input is controlled by subroutine INPRO, the input processor, which calls a different subroutine to read and process each of the different data tables - RDCONT (container), RDLEG (legs), RDSPD (spread functions), RDVEH (vehicles), RDFAC (facilities), RDCRG (cargo elements), RDMISS (missions), and RDRPT (report or output requests). These routines establish the basic data tables; in particular, RDMISS constructs the initial cargo table, representing all cargo to be shipped according to the mission data.

Cargo assignment is controlled by LEGPRO, the leg processor. Sorting the data so that all cargo to be shipped on a given vehicle, leg, and year is grouped in one block, LEGPRO calls subroutine ASINER to determine the actual shipping schedules. LEGPRO augments the cargo table as necessary to provide for additional cargo shipments not specified by mission data, such as bulk containers, crew capsules, propellants, and vehicles required to carry cargo. LEGPRO also transforms the augmented cargo table into a phase II cargo table reflecting the shipping schedules, to be used for output.

Output is controlled by REPORT, the report generator. For each of the reports which has been requested through input, one of the printout subroutines is called. These subroutines draw on information tables established during the leg processing.

On the next page is a diagram of the flow between subroutines. Five small utility subroutines (PACK, UNPACK, READER, SORT and VALUE) are not shown on the Hierarchy diagram.On succeeding pages is a brief description of each subroutine, followed by detailed explanations of each routine with flow charts. A separate section is devoted to explanations of program variables and major tables.



SUBROUTINE HIERARCHY

(Pack, Unpack, Value, Reader, Sort Not Included)

PURPOSES OF ALL SUBROUTINES

ASINER	Controls the cargo assignment process: determines the number of flights for each vehicle on each leg in each year and schedules cargo on those flights.
CNTRPT	Prints the container report - a summary of container usage by year, leg, and name.
COLECT	Shortens the table of vehicle flights.
CSTPRT	Prints the vehicle and facility cost reports.
DDBSFT	Internally shifts some input data in core to allow insertion of more facility or cargo element data.
DORCA	Main routine which calls the input processor, the leg processor, and the report generator.
DPAGER	Used by CSTRPT to improve the appearance of the cost report printout.
FACNUM	Counts the facilities acquired in each year by program, mission, and cargo element.
FACRPT	Prints a report on facility acquisitions in each year by program, mission and cargo element.
FIND	Used in the cargo assignment process to schedule cargo of a specific kind in a specific direction not exceeding vehicle weight and volume and other constraints.
FINDSP	Given the name of a spread function, finds the location of data for this function in the input spread table in DDB or prints a message if no such function was input.
INPRO	Calls the proper subroutines to process input data.
LEGPRO	Processes all cargo data from phase I into phase II, keeping tract of container, propellant and other requirements.
L2WO	Writes the Level II cargo table onto an external file in the proper format.
MERGE	Performs an out-of-core sort on the level II cargo table stored on an external file.
MOREL2	Reads a section of the Level 2 cargo table into available core from the tape on which it is stored.
PACK	Inserts bits from a right-adjusted integer into a given portion of a 36-bit word.
PERLNK	Calculates performance data for a vehicle on each leg if that data was not input.
PROLNK	Sets up an array of data for use by PROPCL and computes off-loaded propellant.

PROPCL	Computes propellant required by a vehicle carrying a given up weight and down weight.
RDCONT	Controls reading and storage of container input data.
RDCRG	Reads and stores cargo element data.
RDFAC	Reads and stores facility data.
RDLEG	Reads and stores leg data.
RDMISS	Reads and stores mission data as the phase I cargo table and also initializes the vehicle and facility acquisition tables.
RDRPT	Reads and stores input requests for reports.
RDSPD	Reads and stores spread functions.
RDVEH	Reads and stores vehicle data and finishes construction of leg table.
READER	Performs actual reading of input cards and performs certain types of processing.
REPORT	Calls appropriate subroutines to provide the reports requested by the user.
SEG22	Calls TRAFIC.
SEG32	Calls MERGE.
SEG33	Calls SPRINT.
SEG36	Calls TRAFIC
SORT	Superfast matrix sorting routine.
SPDAP	Given a spread function, a unit cost, and a count by year of units for a given vehicle or facility, computes the spread costs for each year.
SPRINT	Prints a summary of cargo traffic in detail.
SWITCH	Retains cargo elements and facilities.
TABLES	Prints intermediate and debugging information.
TRAFIC	Determines the actual number of each vehicle type needed to perform the flights generated, based on maximum flights per year, lifetime in years, lifetime in flights, and other requirements.
UNPACK	Extract bits from a given portion of a word and store as a right-adjusted integer.
VALUE	Converts an input numeric value from alphanumeric (Hollerith) representation to machine floating point format.
VEHLDF	Sums up load factors for each vehicle by program, mission, and year.
VEHRPT	Prints a vehicle utilization report in terms of load factors for each year, program and mission.
YEARS	Creates the array NTYRS containing the years (last 2 digits) in which there is nonzero activity (costs incurred, cargo shipped, etc.)

SECTION 2
SUBROUTINE ASINER

Given a vehicle and a list of all cargo items traveling in either direction in a given year on a given leg, ASINER with the help of subroutine FIND determines the number of flights of that vehicle necessary to carry all required and possibly some optional cargo and constructs a schedule of flight assignments. Bulk containers and crew capsules are requisitioned and accounted for as necessary. "Required" (or "regular") cargo is that which must travel on the specified vehicle. "Optional" cargo is any cargo in the capture bin for that leg and year, which may travel on any vehicle it can fit into on that leg.

USAGE

The cargo assignment routine is called by the leg processor, LEGPRO, which has set 4 arguments in common:

COMMON/ARGS/ ICF, ICL, IF, IL

where ICF and ICL indicate the beginning and end of the capture bin for this leg/year in the Level I cargo table, and IF and IL indicate the beginning and end of the required cargo for that leg/year/vehicle. As defined elsewhere, the cargo table is assumed to consist of triplets of data words stored consecutively, each triplet defining one cargo item. IF is the location of the first word of the first item; IL is the location of either the first, second or third word of the last item. Similarly for ICF and ICL.

The results consist of three variables and three matrices, which are stored in labeled common/ASDAT/ and which are defined below:

NFLT - total number of flights scheduled.

TW(I, J) - total weight carried in up-direction (I=1) and downwards (I=2) on flight number J (J=1, 2, . . . , NFLT), including extra bulk containers requisitioned. A negative down weight for any flight indicates that the vehicle is to be expended (does not return).

NASS - total number of assignments made in both directions in the FLTA matrix.

FLTA(I, J) - assignment information for cargo item J, J=1, 2, ..., NASS. The entry for I=1 is a packed word containing four pieces of information:

Bits (number)		Contents
0-15	(16)	Subscript IS of this cargo item in the cargo table (DDB). ICF $\leq$ IS $\leq$ ICL or IF $\leq$ IS $\leq$ IL.
16-24	(9)	Flight number N to which item is assigned. $1 \leq N \leq$ NFLT.
25	(1)	Direction (0-up, 1-down).
26-32	(7)	Subscript K in CR array indicating container in which item is stored (used only for bulk material or crew, zero for discrete items).

The entry for I=2 contains the weight of item J which has been assigned. For bulk items which have been divided among several flights or containers, this weight is less than the original weight in the cargo table.

NCR - number of containers requisitioned and listed in CR array.

CR(I) - packed word giving information on the I^{th} bulk container requisitioned for these flights, I=1, 2, ..., NCR.

Bits (number)		Contents
0-17	(18)	Index in container table TBCØNT pointing to data for this kind of container.

18-26	(9)	Up flight number.
27-35	(9)	Down flight number.

A bulk container carries bulk cargo in only one direction and flies empty on the return trip. To determine which items are stored in container I, check bits 24-30 of the items in the FLTA matrix to find those whose container subscript equals I.

LIMITATIONS

For each vehicle/leg/year combination, the following limits are currently in force:

Number of flights	-	100 (due to dimension of TW matrix).
Number of Assigned Items	-	400 (due to dimensions of FLTA and WS matrices). Note: items traveling round trip account for two assignments. Bulk cargo split into several portions and stored in several containers account for one assignment for each portion.
Number of Containers Requisitioned	-	100 (due to dimension of CR list). The up and down trips for a single bulk container constitute only one entry in CR, while for a crew capsule usually one but sometimes two entries are used.

The following is a detailed explanation of cargo assignment procedures.

VEHICLE CAPABILITY

The vehicle capability is presently represented by five input data items:

3

LIMVEH	-	maximum number of cargo items which travel in each direction on any flight (deployment limit).
UPMAX	-	maximum weight the vehicle can carry upwards (away from the earth) if it returns empty.
DNMAX	-	maximum weight the vehicle can carry downwards if it travels upwards empty.
EXMAX	-	maximum weight the vehicle can carry upwards if it is expended (does not return).
VOLMAX	-	maximum volume capacity.

The less a vehicle carries upwards, the more it can carry downwards, and the relationship expressing down capacity as a function of up load weight is a curve which is here taken as a straight line. Since usually UPMAX is greater than DNMAX, for every pound the vehicle carries down, it loses more than a pound in upward capacity. To express this fact, we deal with the "equivalent up weight" of every cargo item. For a downward-bound item, the equivalent up weight is the actual weight times the factor $\frac{\text{UPMAX}}{\text{DNMAX}}$, which is usually greater than 1. For an upward-bound item, of course, the equivalent up weight equals the actual weight. During the assignment process, the program keeps track of the remaining unused vehicle capacity for each flight through the variable VCAP, which is expressed in equivalent up weight. For each flight, the program continues loading the vehicle until VCAP drops below the CUTOFF value (presently 10), at which point the vehicle is considered fully loaded, and the next flight is set up.

The program also keeps track of the remaining volume on the vehicle in both directions through the variables RVOL (1) and RVOL (2). In many cases the program will schedule a vehicle which is less than fully loaded because, at a certain point in the algorithm, no permissible cargo remained that did not exceed remaining vehicle capacity in weight or volume or violate some other constraint.

A deployment limit may be input for the vehicle, the leg, or both (in which case the lower limit is used for the vehicle/leg combination). Each discrete cargo item counts once for the direction it travels; each crew capsule or bulk container plus all its contents counts once. Items which are packed inside a capsule or container do not count separately.

CAPTURE BIN

"Optional" cargo from the capture bin is assigned to a vehicle only to fill up space into which no remaining regular cargo can fit. This may be either

(1) Bulk cargo to fill a partially empty bulk container or crew capsule.

(2) A crew, discrete item(s) or containered bulk.

All "optional" cargo must satisfy all existing flight restrictions: weight, volume, round trip/same vehicle, deployment limits, and so on.

To simplify program bookkeeping, if a portion of a bulk cargo item from the capture bin is assigned, then the remainder of that item is removed from the capture bin and relabeled as regular (mode 3) cargo. This may sometimes result in generating additional flights to handle the remaining bulk, but to avoid that the bookkeeping problems would be horrendous.

Upon termination of the algorithm and exit from ASINER, some cargo will often remain unassigned in the capture bin. Frequently, the routine will assign only one direction of an item travelling round trip (unless the round-trip-same-vehicle flag is set for those items). These unassigned items are in WS(I, J), J = ITEMS, ..., MAXI (See section on WS).

CARGO CLASSIFICATIONS

Each cargo item is tagged with numbers representing three or more sets of classification, which are:

MODE	=	1	Cargo items requiring an expended vehicle (weight exceeds UPMAX but does not exceed EXPMAX).
		2	Optional cargo from the capture bin.
		3	Regular cargo required to travel on the vehicle specified.
TYPE	=	1	Manned crew, which requires a crew capsule.
		2	Bulk cargo, which can be subdivided as necessary to fit into unused space and which must be stored in a container.
		3	Discrete cargo, which is self-contained and cannot be subdivided.
DIRECTION	=	1	Upwards (away from earth).
		2	Downwards (toward earth).
CONTAINER INDEX	=	N	for crews, where entry #N in the container table is a crew capsule.
		N	for bulk cargo, where entry #N in the container table is the required bulk con container.
		0	for discrete items.

These tags are packed with other information for each item into a working storage matrix WS. They are used to load the vehicles in a fairly efficient and practical manner.

ASSIGNMENT OF CREWS

At most one crew may be assigned to a flight in each direction. Assigning a crew requires assigning a crew capsule, also. The total weight is thus the crew weight plus capsule weight; the total volume is the capsule volume. A crew capsule has a given capacity (input), expressed in terms of weight. Any excess capacity left over after the crew weight has been subtracted may be used to carry uncontainered bulk material.

ASSIGNMENT OF BULK CARGO

Bulk cargo must be shipped in a container, which may be either a crew capsule with excess capacity or a bulk container. The user may, if he wishes, broadly separate all bulk cargo into several general kinds, each of which requires its own special kind of bulk container. Any kind of bulk may be loaded into the crew capsule except propellant, and in fact several different kinds may be shipped together in a capsule if space permits. However, a bulk container may carry only bulk material which is specifically designated for that kind of container.

Bulk cargo has no specific volume of its own. However, the container in which it is shipped must not exceed the remaining volume capacity of the vehicle in that direction.

Unlike crew and discrete cargo items, bulk cargo can be subdivided into smaller parcels as necessary to fit into available spaces. Whenever a crew capsule or bulk container is to be filled, the program searches for an acceptable bulk item whose weight does not exceed the remaining space; if none exist, the program will bite off a piece of a large bulk item to exactly fit the remaining space. The term "remaining space" means the smaller of remaining container capacity and remaining vehicle capacity.

The program attempts to load as much bulk material into crew capsules as possible, to minimize the number of additional bulk containers required. In doing so, it loads first those kinds of bulk material which remain in smallest quantity; this policy is designed to avoid insofar as possible having to load a fairly heavy container for a relatively small amount of material. When capsule space is exhausted and bulk cargo remains unassigned, the program must requisition a bulk container. Subject to weight, volume, and container capacity limitations, the program chooses to load next that kind of bulk material which can be loaded in greatest quantity (weight), which is another attempt at trying to avoid loading bulk containers which are almost empty.

When a vehicle is almost full in terms of weight, there is an inherent danger that the algorithm will assign a bulk container and then find that the amount of bulk that can be carried within the vehicle weight limits is uneconomically small. For example, if remaining vehicle capacity is 5300 units and a bulk container weighs 5000 units, only 300 units of bulk could be carried. To avoid such uneconomical arrangements, ASINER will not load a bulk container if remaining vehicle capacity (after subtracting the container empty weight) is not sufficient to load at least X percent of the container capacity. This X percent, denoted by the variable BLKLIM, is now set to 20%. Thus, a bulk container weighing 5000 units and with a capacity of 40000 units cannot be loaded unless remaining vehicle capacity is at least 13000 units (8000 minimum for bulk, 5000 for the container). This restriction does not apply if only a small amount of bulk cargo is left.

The user has a choice, through input, of how to handle propellant. One way is to simply designate it as bulk material, indicating in the container table a container specifically branded as a propellant tank. The program will handle the propellant like any other kind of bulk except that propellant will not be stored in the crew capsule. However, a bulk container may be shipped only partly full if vehicle capacity is used up before the container is full. Thus, if the user wishes to ship only full tanks of propellant (or any other type of bulk), he should input it as self-contained discrete items whose up weight is the weight of propellant plus tank and whose down weight is the weight of the empty tank.

The CD matrix is used to keep track of how much regular bulk of each kind remains unassigned. Before the first assignment is made, CD(I, J) = the total (weight) of all cargo of type J remaining in direction I (1-up, 2-down), where J refers to the container table. As bulk cargo is assigned, the weight is subtracted from the proper slot in CD. CD2(I, J) is used similarly to keep track of mode 2 bulk cargo from the capture bin.

CONTAINER REQUISITIONS

Every time a crew capsule or bulk cargo container is assigned, a new entry is made in the CR list, so that other portions of the program can account for containers needed and available. Each entry in the CR list contains 3 pieces of information packed into a single word:

1) A pointer J to the container table to indicate which kind of container has been assigned.
2) The flight number of the up trip.
3) The flight number of the down trip.

Bulk containers travel filled in only one direction and, if container return is requested, are shipped back empty. If space and weight limits permit, a bulk container makes the round trip on the same flight (up and down flight numbers are the same). Otherwise, a new entry in the WS matrix is created for the return of the empty container, being designated as discrete item. If containers are to be expended, this procedure is skipped.

Crew capsules are not automatically returned; they are returned only if a crew is scheduled for return. If a crew capsule makes a round trip on the same flight, it will give rise to a single entry in the CR list. If the round trip is on different flights, two entries will be created in CR; one entry represents the up-flight and has a down-flight number of zero, while the second entry represents the down-flight and has an up-flight number of zero.

ASSIGNMENT RULES

The program makes assignments of cargo items from various categories in a certain order, as set forth below. The category tag values (MODE, TYPE, DIRECTION, CONTAINER INDEX) are set in subroutine ASINER along with the deployment limits and maximum weight and volume which can be assigned, and subroutine FIND then searches the working storage array for items satisfying the following specifications:

(a) The item must be of the indicated MODE, TYPE and DIRECTION.

(b) If bulk is requested, the container index must match the specified container index MCT.

(c) The equivalent up-weight of the item (plus its container, if crew or bulk) must not exceed the remaining vehicle capacity VCAP.

(d) The volume of the item (if discrete) or its capsule/container (if crew or bulk) must not exceed the remaining vehicle volume RVOL(I) in that direction.

(e) If the item requires single deployment, no other item may be assigned to this flight in this direction. If multiple deployment is acceptable, the total number NOCC (I) of items assigned in direction I must not exceed the limit LIMOCC(I), which may be imposed either on the vehicle or on the leg. The program sets LIMOCC(2) = 0 for a flight which is to be expended.

(f) If the item must make a round trip on the same flight, the above specifications must be observed in both directions.

The order of assignment for each flight is as follows.

1. On a given flight, if any "expended" cargo (MODE 1) remains, assign one of that first. The rest of that flight can be filled with other kinds of cargo traveling upwards. Mode 1 cargo can be either a crew or discrete cargo but not bulk material, since bulk can be subdivided into smaller pieces not requiring an expended vehicle.

2. Assign a crew upwards, if possible. If the flight is not to be expended, try to assign a crew downwards.

3. Try to fill rest of vehicle with discrete cargo until VCAP (remaining vehicle capacity) falls below CCAP (remaining capsule capacity). CCAP may be zero -- for example, when no capsule has been assigned. Preference is given to up cargo over down cargo.

4. If a manned capsule was assigned, fill it with bulk cargo traveling in the same direction. If crew capsules were scheduled for both directions on this flight, only the up-trip is used to carry bulk. First load regular cargo, starting with that kind which remains in smallest quantity. If all regular (mode 3) bulk is exhausted before the capsule is filled, mode 2 cargo from the capture bin will be assigned. Should all bulk cargo be exhausted before VCAP falls below the CUTOFF value, the program will return to assigning discrete cargo to fill up the vehicle.

5. When no more discrete cargo remains, the program loads bulk material, for which a container must be requisitioned, assigning a new container whenever the current one is filled to within a CUTOFF value of capacity (now 10). Each time a new container is assigned, the program chooses the kind and the direction so as to maximize the quantity that will be loaded into the container. This quantity W is expressed as

$$W = \text{minimum of} \begin{cases} \text{Vehicle capacity minus container weight;} \\ \text{Container capacity;} \\ \text{Total amount of bulk remaining of this kind} \\ \quad \text{and direction.} \end{cases}$$

Each time a new container is loaded, if containers are to be returned (empty), the routine schedules the return of the empty container on the same flight if space permits. If this is impossible, an entry in the WS matrix is created for shipment of the container as an empty discrete item in the opposite direction. If a container is partially unfilled because no more regular (mode 3) cargo remains, the program searches the capture bin for bulk of this kind and direction.

For each flight the routine proceeds as far as possible through steps 1-5 until either

(a) The vehicle is filled on the basis of weight, volume (both directions), or deployment limits; or

(b) No remaining cargo, regular or optional, can fit into the remaining space.

In either case, the flight is now considered "full". If no more regular cargo (mode 1 or 3) remains unassigned, the algorithm terminates and ASINER exists. Otherwise, the next flight is processed in the same manner.

WORKING STORAGE MATRIX WS

The items in the cargo table are reconstituted and stored in the matrix WS for working purposes. Cargo items required to make a round trip give rise to two entries in WS, the first corresponding to the up trip, the second to the down trip. The exception to this is items requiring a round trip on the same flight; only one entry is created in WS, and the up-weight is assumed to equal the down weight. WS (1, J) contains a packed word for entry J with 8 pieces of information:

Bits	(number)	Contents
5	(1)	If = 1, item requires single deployment. If = 0, multiply deployment is acceptable.
6	(1)	If = 1, item must make round trip on same flight. If = 0, round trip may be scheduled on separate flights.
7	(1)	If = 1, this item is a requisitioned container which has already been scheduled to carry bulk in the opposite direction. If = 0, this item originated in the cargo table.
8-11	(4)	Container index (refers to container table TBCONT).
12-29	(18)	If bit 7 is 0, this field contains the subscript IS in the cargo table where this item originated. If bit 7 is 1, this field contains the index I in the CR array giving data on this container.
30-31	(2)	Direction (1-up, 2-down).
32-33	(2)	Type (1-crew capsule, 2-bulk, 3-discrete).
34-35	(2)	Mode (1-expended vehicle, 2-capture bin, 3-regular).

WS (2, J) contains the weight of item J.

Corresponding to entry J in WS is VOL(J), which contains the volume of this item (zero for crews and bulk).

When a cargo item is assigned to a flight, it is deleted from WS and VOL arrays and replaced by the last item in WS, so that the size of WS is reduced by one. When a new container is requisitioned whose return cannot fit on the same flight, a new entry in WS is created for the shipment of the empty container in the opposite direction as a discrete item. Unassigned items are stored in the upper portion of WS. At any time the number of items remaining unassigned in WS is given by the expression MAXI-ITEMS+1.

ERRORS DETECTED

The following list explains error messages printed by ASINER and FIND, their probable sources, and the action taken by the program.

1. "Invalid volume limit for current vehicle." The volume limit was found to be a negative number. Since the input routine RDVEH checks for this also and since the default value is 1000, this problem is not an input error, but rather arises from some blow-up elsewhere in DORCA. ASINER will skip all passes for this vehicle.

2. "ASINER cannot find data for leg on vehicle VVV." Most likely the user simply forgot to input this data. Program will skip this vehicle/leg combination for all years.

3. "Cargo element named XXX has illegal value for container class (KKK)." Means that bits 12-23 of word 6 for this element in the cargo element table do not contain the value 1, 2, 3, or 4. This is an internal programming problem. ASINER will simply skip this element and continue processing other elements.

4. ""'*REJECT*CARGO XXX CONT.CCC VEH. VVV LEG LLL" Means that the container volume exceeds the vehicle volume limit or that the weight of the empty container plus 20% of its capacity exceeds the vehicle weight limit in the down direction. ASINER will skip all cargo elements which must travel in the container indicated. Most likely an input error.

5. "Cargo element named XXX has weight of WWW in direction K; weight must exceed cutoff value of EEE for proper processing." Currently, EEE is taken as 0.99. Most likely an input error, or possibly an instance where the user input a dummy cargo item of very little or no weight for convenience. Program will skip this cargo element.

6 "*REJECT*CARGO XXX WT. WWW VOL. YY VEH. VVV LEG. LLL" Either a discrete item whose volume exceeds the vehicle volume capacity or a discrete or crew cargo item whose weight exceeds the weight capacity of the vehicle in the direction(s) the cargo is traveling. Input error. Program ignores this cargo element.

7. "Too many cargo items on vehicle VVV, leg LLL." This is due to a large amount of data which exceeded the dimensions of matrices FLTA and WS. Probably due simply to a lot of cargo, in which case the problem can be alleviated by increasing the dimensions of FLTA and WS or be rearranging the data somehow to reduce the cargo. Could be due to an input error, such as a bulk container with a ridiculously small capacity. ASINER may have already made assignments for some of the cargo before the problem arises. If the problem was detected in ASINER, it will immediately discontinue processing the current vehicle/leg/year and go on to the next one; if the problem was detected in subroutine FIND, it will abort the entire program.

8. "Too many flights of vehicle VVV on leg LLL. Outgrew TW matrix." Means that the dimension of matrix TW was exceeded. Program will discontinue the current vehicle, leg and year and go on to the next. To solve this problem, increase the dimension of TW and also the value of variable MAXF, which contains the dimension of TW.

9. "ASINER found container named CCC has inadequate capacity." Means capacity was found to be zero or negative. Since the input routine RDCONT also checks for this, this message indicates that the data were destroyed somehow and that a programming error exists somewhere. Program will abort immediately.

10. "Too many containers requisitioned by ASINER for vehicle VVV on leg LLL. Limit is MMM." Outgrew CR list. Indicates that dimension of CR array was exceeded. Program will abort immediately. A possible solution is to increase the dimension of CR and the value of variable MAXC. One should also carefully examine the input data, since this problem could arise from an input error such as a container with a ridiculously small capacity or an astronomical amount of bulk material to be shipped.

11. "Cargo ASINER is apparently in an infinite loop, having made consecutive unsuccessful calls to subroutine FIND." Means that the cargo items remaining unassigned do not match the records, so that ASINER is searching for something that does not exist. Caused by some internal programming error. Program will dump all of core, then abort immediately.

DEFINITION OF VARIABLES USED BY SUBROUTINE ASINER AND FIND

Variable	Definition
BLKLIM	This variable (now set to 20%) when applied to a bulk container's capacity indicates the minimum load in pounds which may be shipped in a bulk container. This restriction does not apply if the total remaining amount of bulk cargo of this kind is itself less than this limit. This policy is designed to avoid requisitioning a weighty bulk container for an almost-full vehicle to carry small amounts of bulk unless only a small amount is left.
CCAP	Amount of container capacity remaining unused. The container may be either a bulk container or a crew capsule with extra space. This quantity does not depend on direction.
CD(I, J)	Total amount of bulk cargo still unassigned in direction I (I = 1,2) for container type J (J = 1, ..., NCONT), mode 3 cargo.
CD2(I, J)	Same as CD but for mode 2 cargo.

CR	Array giving list of containers requisitioned, both bulk and crew. CR(I) contains a packed word giving information on a single container, I = 1, 2, . . . , NCR.

Bits 0-17 (18 bits) - Index in container table pointing to data for this container type.

Bits 18-26 (9 bits) - Up flight number.

Bits 27-35 (9 bits) - Down flight number.

Bulk containers are assigned to carry bulk cargo in only one direction. To find which items are stored in container I, check bits 26-32 of the items in the FLTA matrix to find those whose container index is I.

The round trip for a bulk container constitutes only one entry in CR. The same is true for crew capsules making a round trip <u>on the same flight</u>. Crew capsules making a round trip on different flights give rise to two entries; for one entry the up-flight number is zero, for the other entry the down-flight number is zero. In this case the capsule is considered to be two capsules, one going up only, the other going down only.

CONTRE	Global container return option. If CONTRE = 0, empty bulk containers are returned. If CONTRE $\neq$ 0, containers are expended.
CUTOFF	When unused vehicle capacity (VCAP) drops below this small tolerance (currently 10), vehicle is considered to be filled and next flight is started. When unused container capacity CCAP drops below CUTOFF, current container (bulk or crew) is considered to be full, and a new container must be assigned for more bulk cargo.
CW	Container weight.

C1, C2	Name of cargo item (A6, A4 format).
D	Contains coded word "UP" or "DOWN" for error printout.
DDB	Long array containing various kinds of data, including the vehicle input data, the cargo element descriptions, and the cargo table processed by ASINER.
DIRECT	Direction of travel. 1-up (away from earth), 2-down (towards earth).
DNMAX	Maximum weight which vehicle can carry downwards if it flies upwards completely empty.
EXP	Contains coded word YES or NO indicating whether current flight is scheduled to be expended (vehicle does not return).
EXPMAX	Maximum weight which vehicle can carry in expended mode (vehicle does not return at all).
FACTOR	FACTOR(I) contains the equivalent up-weight factor for this vehicle in up direction I (I = 1) and down (I = 2). FACTOR(1) = 1., FACTOR(2) = UPMAX/DNMAX.
FLAG	If nonzero, indicates that another container has just been assigned and is to be entered into the CR list and accounted for in other running totals.
FLTA	Matrix containing the assigned items, FLTA(I, J), I = 1, 2; J = 1, 2, . . . , NASS. FLTA(1, J) contains a packed word with information for item J:

Bits 0-15 (16 bits)	- subscript IS of this item in the cargo table (DDB array). IF ≤ IS ≤ IL.
Bits 16-24 (1 bit)	- Flight number N to which item is assigned. 1 ≤ N ≤ NFLT.
Bit 25 (1 bit)	- Direction (0-up, 1-down).
Bits 26-32 (7 bits)	- Subscript K in CR array indicating the container in which bulk cargo or crew is stored. (Zero for discrete, self-contained items.)

FLTA (2, J) contains the weight of item J which has been assigned. If item J is bulk material, this weight may be less than the original weight specified in the cargo table if the item has been divided among more than one flight or container.

FLTA shares core space with the WS matrix. As items are assigned, they are shifted from the upper end of this space to the lower end. Indices are controlled so that no overlapping occurs.

I	Scratch variable.
ICE	Cargo element number.
ICF	Beginning of capture bin for this leg/year in DDB.
ICL	End of capture bin for this leg/year in DDB.
ICT	Kind of container required by current bulk cargo or crew item (corresponds to index in CD and TBCONT arrays).
ID	Direction of current cargo item. See DIRECT.
IF	Index in cargo table (DDB array) of first regular cargo item for this vehicle/leg/year.
IL	Index in cargo table (DDB array) of last regular cargo item for this vehicle/leg/year.
IM	Mode of item currently being processed. See MODE.
INDEX	If nonzero upon return from subroutine FIND, indicates that a cargo item satisfying required mode, type, direction, container type, weight limit and volume limit was found and assigned to current flight. If zero, no such item could be found. The absolute value of INDEX indicates the location in the WS matrix where the item was stored (may now have been replaced by another item). If INDEX < 0, a bulk item was split up and only a portion used.

IP	Scratch variable.
IRELDT	The year to which all dates are relative (ie - 1970).
IRT	1-bit round-trip flag in cargo table. If found to be equal to 1, cargo item must make round trip. If equal to zero, item is traveling in only one direction, which is indicated by the ID bit (0-up, 1-down).
IS	Subscript of current cargo item in DDB array.
ISD	$= \begin{cases} 0 & \text{if multiple deployment is o.k. for this cargo} \\ 1 & \text{if single deployment is required} \end{cases}$
IT	Type of current item. See TYPE.
ITEMS	Lower index limit on unassigned items remaining in WS/VOL matrices. Actual number of items remaining unassigned at any instant is MAXI + 1 - ITEMS.
IX	Scratch variable used in masking operation.
IYR	Year of cargo shipment, relative to IRELDT.
J	Scratch variable.
JCE	Location of beginning of data for this cargo element in DDB.
JDONE	$= \begin{cases} 0 & \text{if some cargo is still unassigned} \\ 1 & \text{if all regular and optional cargo has been assigned} \end{cases}$
JERR	Total number of errors discovered in program.
J1, JL	Lower and upper limits of DDB array containing data for current vehicle.
JRT	$= \begin{cases} 1 & \text{if cargo must travel round trip on same vehicle flight} \\ 0 & \text{otherwise} \end{cases}$
K	Scratch variable used in several senses.
KVEH	Number of current vehicle.

L — Scratch variable.

LEG — Number of current leg.

LIMCAL — Limit on total number of consecutive unsuccessful calls to subroutine FIND before program aborts itself. Used to prevent infinite loops due to some undetected input error.

LIMLEG — Deployment limit for any vehicle on this leg.

LIMOCC(I) — Deployment limit for this flight up (I=1) and down (I=2). If vehicle is to be expended, LIMOCC(2) = 0. Otherwise, LIMOCC(I) = min (LIMVEH, LIMLEG).

LIMVEH — Deployment limit for this vehicle on any leg.

MANNED — Indicates whether a manned capsule has already been assigned to this flight and in what direction.

$$\text{MANNED} = \begin{cases} 0 & \text{if no crew has been assigned.} \\ 1 & \text{if crew has been assigned upwards only.} \\ 2 & \text{if crew has been assigned downwards only.} \\ 3 & \text{if crew has been assigned both ways.} \end{cases}$$

MAXA — Maximum number of assignments which can be listed in FLTA matrix for current leg, vehicle and year. Items making round trips constitute two assignments; bulk cargo which is split up into several portions and stored in several containers constitute several assignments.

MAXC — Maximum number of requisitioned containers for a given leg, vehicle, year.

MAXF — Maximum number of flights which can be totaled in TW matrix.

MAXI — Maximum number of cargo items which can be stored in WS array at any time.

MBASE — MBASE=2 if all regular cargo was assigned and last flight is being filled from capture bin.
MBASE=3 if some regular cargo remains unassigned.

MCBULK	Set to 1 when a manned capsule is assigned with some space for bulk cargo. Reset to zero after that space is filled.
MCHG	If nonzero, indicates that a portion of bulk cargo was removed from the capture bin and redesignated as regular cargo (mode change).
MCT	Index in CD and TBCONT matrices indicating what kind of bulk cargo is to be loaded now, if more than one kind exists.
MODE	Classification of cargo items according to treatment: $\text{MODE} = \begin{cases} 1 & \text{for items requiring an expendable vehicle.} \\ 2 & \text{for priority items.} \\ 3 & \text{for regular items.} \end{cases}$
MTD	Packed word containing required mode, type and direction for next assignment.
N	Scratch variable indicating number of cells in DDB used for data for current vehicle.
NASS	Number of items already assigned to vehicle flights and stored in the FLTA matrix.
NB	Scratch variable indicating bit position for packing and unpacking data in CR array.
NBCE	Index in DDB array indicating beginning of block of data describing the cargo elements.
NCALL	Number of consecutive unsuccessful calls to subroutine FIND.
NCONT	Number of different kinds of containers (bulk or crew capsule) in container table TBCONT.
NCR	Number of containers requisitioned and listed in CR array.
NEV	Number of expended vehicles required for this leg and year.

NFLT	Number of flights scheduled for this vehicle on this leg in this year.
NMODE(I)	Number of items of mode I remaining unassigned.
NO	Contains coded word "no" for comparison with EXP.
NOCC(I)	Number of self-contained items loaded on this flight so far in direction up (I=1) and down (I=2).
NWCE	Number of words in each cargo element description.
NWVEH	Number of words of basic data in vehicle data table for each vehicle.
RVOL(I)	Remaining vehicle volume capacity on current flight in up direction (I = 1) and down direction (I = 2).
TBCONT	Container data table.

TBCONT(I, J), I = 1, ..., 8; J = 1, ..., NCONT

Container type # J is described by the 8 words of column J:

Words 1-2:	Name (A6, A4 format).
Word 3:	Capacity (lbs).
Word 4:	Container empty weight (lbs).
Word 5 =	1 for crew capsules, 2 for bulk containers, 4 for propellant tanks.
Word 6:	Volume factor.
Word 7:	Used in CNTRPT
Word 8 =	0 to return empty bulk containers, 1 to expend containers.

TBLEG	Leg data table. See full description in Appendix B.
TBVEH	Vehicle table. See full description of vehicle data in Appendix B.

TCAP	Temporary variable containing remaining unused container capacity.
TOTAL	Total amount of whichever kind of bulk cargo remains in smallest supply in the same direction as the manned capsule to be filled. Used to determine which kind of bulk to load.
TRIFLE	Cutoff value to determine when certain quantities have been exhausted. Currently set to 0.99.
TVCAP	Temporary variable for remaining vehicle capacity.
TW	TW(I, J) gives the total weight to be carried by the vehicle on flight number J ($1 \le J \le$ NFLT) in direction I (1-up, 2-down)
TYPE	Classification of cargo items by storage requirements:

$$\text{TYPE} = \begin{cases} 1 & \text{for crews which require a crew capsule.} \\ 2 & \text{for bulk cargo requiring a container for shipment.} \\ 3 & \text{for discrete items which are self-contained.} \end{cases}$$

UPMAX	Maximum weight which vehicle can carry upwards, assuming vehicle returns empty.
VCAP	Amount of vehicle carrying capacity remaining unused, expressed as equivalent up-weight.
VDONE	Set to 1.0 when entire vehicle capacity is full for this flight.
VOL(I)	Volume of unassigned cargo item I (ITEMS $\le$ I $\le$ MAXI). This array accompanies matrix WS and is indexed the same.
VOLMAX	Maximum volume capacity of current vehicle.
VOLUME	Volume factor of current cargo item.
V1, V2	Name of current vehicle (A6, A4 format).

W	Item weight.
WEIGHT	Equivalent up-weight of cargo plus its container (if any) on a round trip.
WLEFT	WLEFT(I, J, K) = total weight of still-unassigned cargo of mode I, type J, direction K.
WMAX	Weight of that kind of bulk cargo which remains unassigned in greatest quantity in either direction.
WS	Working storage matrix containing two words for each cargo item to be assigned in each direction in which it must travel. WS(1, J) contains a packed word for item J (ITEMS $\leq$ J $\leq$ MAXI) with 8 pieces of information:

Bit 5 (1 bit)	-	If zero, multiple deployment acceptable. If nonzero, cargo item requires single deployment.
Bit 6 (1 bit)	-	If nonzero, this item must travel round trip on same flight. If zero, either not round trip or else separate flights acceptable.
Bit 7 (1 bit)	-	if nonzero, indicates that this item is a requisitioned container which has already been scheduled and filled in the opposite direction with bulk cargo. If zero, this item originated in the cargo table.
Bits 8-11 (4 bits)	-	Index in CD matrix indicating kind of bulk container required (for bulk cargo or crew only).
Bits 12-29 (18 bits)	-	If bit 7 is zero, this field contains the subscript IS in the cargo table where this cargo item originated. If bit 7 is nonzero, this field contains the index I in the CR array giving data on this container as scheduled in the opposite direction.

Bits 30-31 (2 bits) - Direction (1-up, 2-down).

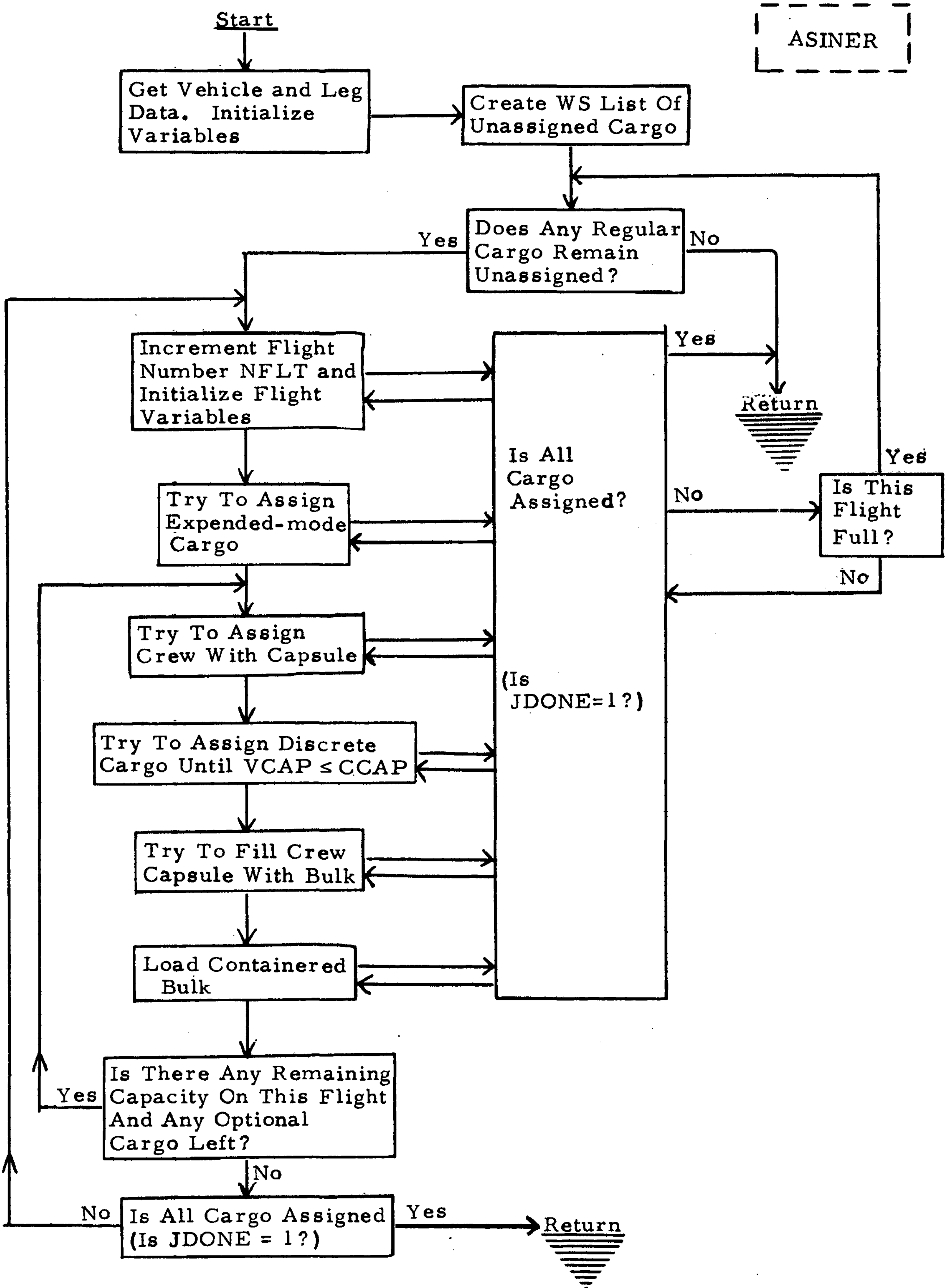
Bits 32-33 (2 bits) - Type (1-crew capsule, 2-bulk cargo, 3-discrete cargo).

Bits 34-35 (2 bits) - Mode (1 - expendable, 2-optional, 3-regular).

WS (2, J) contains the weight of the item. Array VOL(J) contains accompanying volume and is indexed the same.

FLTA shares core space with the WS matrix. As items are assigned, they are shifted from the upper end of this space to the lower end. Indices are controlled so that no overlapping occurs.

WT	Item weight
WTLIM	Maximum weight of any item to be assigned, taking into account the unused vehicle capacity, the direction, and container weight if it is necessary to add another container.
WTMAX	Weight of the largest unassigned cargo item found so far which satisfies mode and other specifications.
X	Scratch variable.
XL1, XL2	Name of current leg (A6, A4 format).
Y	Scratch variable.
YES	Contains coded word "YES" for comparison with EXP.


ASINER
Start
Get Vehicle and Leg Data. Initialize Variables
Create WS List Of Unassigned Cargo
Does Any Regular Cargo Remain Unassigned?
Yes
No
Increment Flight Number NFLT and Initialize Flight Variables
Try To Assign Expended-mode Cargo
Try To Assign Crew With Capsule
Try To Assign Discrete Cargo Until VCAP ≤ CCAP
Try To Fill Crew Capsule With Bulk
Load Containered Bulk
Is All Cargo Assigned?
(Is JDONE=1?)
Yes
Return
No
Is This Flight Full?
Yes
No
Is There Any Remaining Capacity On This Flight And Any Optional Cargo Left?
Yes
No
Is All Cargo Assigned (Is JDONE = 1?)
No
Yes
Return

SECTION 3

SUBROUTINE CNTRPT

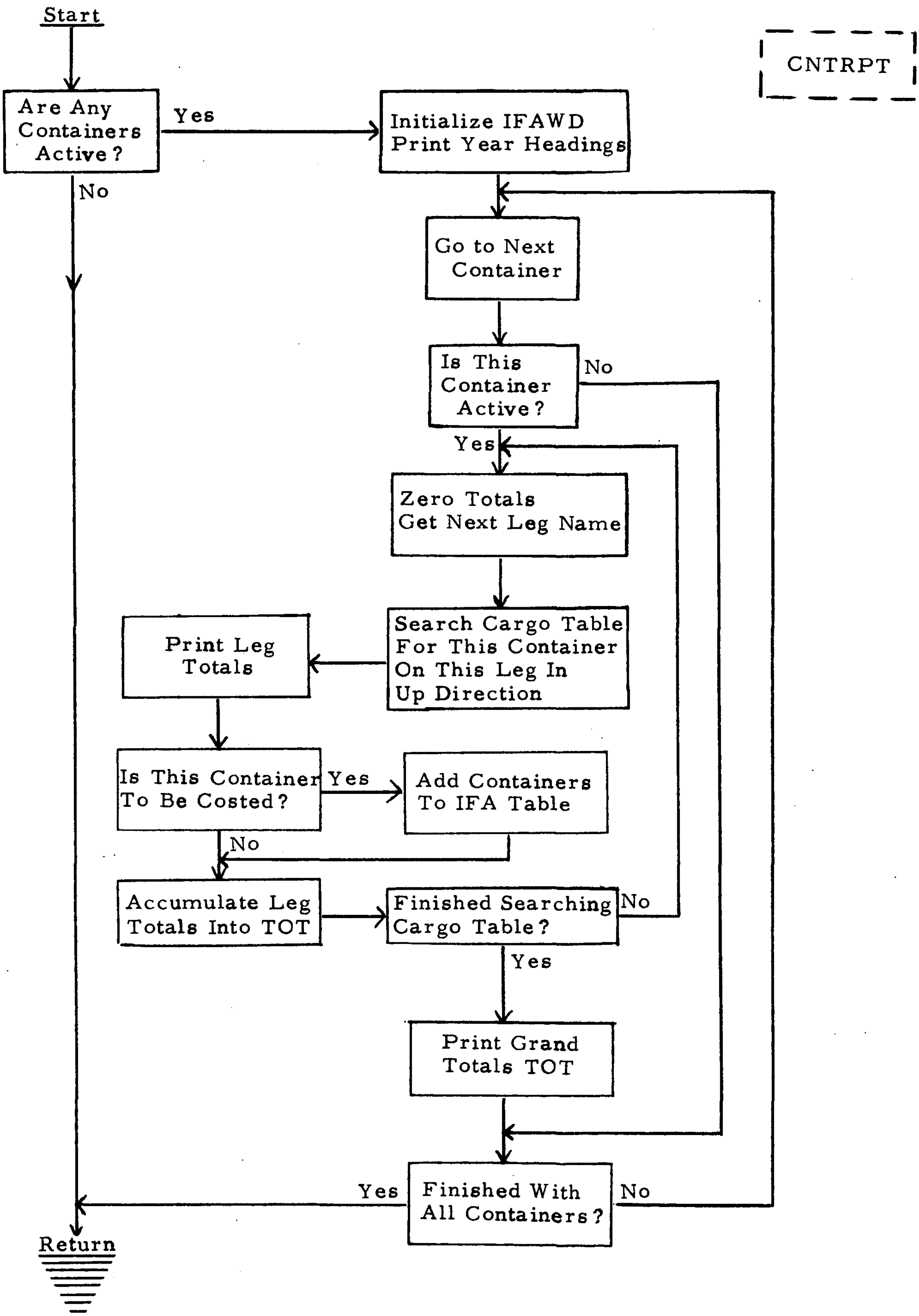
CNTRPT prints a summary of the number of containers of each kind used on each leg in each year. Totals are also provided for those leaving the earth's surface only. This is an optional report which can be requested by input.

All containers were added to the cargo element table in DDB by input routine RDCRG and form a block of cargo elements with indicies I ranging from NBCNT to NCE.

Under the present algorithm, each container used to carry cargo in one direction is either expended or scheduled for a return trip empty (in the opposite direction). Thus, to avoid duplication, the routine counts only containers travelling upwards.

If container #I has indeed been used on the current case, LEGPRO sets TBCONT(7, I) = 1; if inactive, TBCONT(7, I) = 0. For each container I with nonzero activity the routine searches the entire cargo table. For every item in the cargo table, CNTRPT extracts the leg number ILEG, index ICE referring to the cargo element table, direction IDIR, and year of shipment IYR. If ICE = I, ILEG = current leg JLEG, and IDIR = 0 (upwards), then the count of units for year IYR, stored in array LINE, is incremented by 1. If ICE $\neq$ I or IDIR $\neq$ 0, this item is ignored, and the routine proceeds to the next cargo item. If ILEG ILEG $\neq$ JLEG, then all cargo items for current leg JLEG have been processed. If the total in LINE(3) is greater than zero, indicating that at least one container of type I has been shipped on this leg, the leg name and yearly totals are printed. If in addition this leg originates at earth's surface (as indicated by the fact that the name of the next lower leg is "NONE"), leg JLEG is set to the next leg ILEG, the array LINE is zeroed, and the name of the next leg is inserted into the first two words of LINE. Processing continues in this manner until the location L in the cargo table DDB exceeds its upper limit NLDDB, which completes the work for container I. Now container I + 1 is processed similarly.

In order to permit costing of containers, DORCA employs a device which uses the facility costing algorithm. If the user wishes cost reports on one or more containers, he should input to the facility table dummy facilities whose names are the same as the containers to be costed. The program recognizes this situation. In the cargo element table subroutine RDCRG sets a nonzero pointer (which points to the dummy entry in the facilities table) for each such container, whereas the pointer is zero for containers not to be costed. CNTRPT checks this pointer, called NDEX. If NDEX $\neq 0$, CNTRPT generates entries in the facility acquisition table (IFA) for the number of containers acquired in each year. The program/mission are automatically taken to be OVERHEAD/ CONTAINERS (program #1, mission #2). The packed word IFAWD is used to generate the IFA entries, in the prescribed format. The decimal number 66 packed into the first 12 bits represents the program and mission numbers (6 bits each). Since only 6 bits are allocated for the number of units acquired, CNTRPT must make separate entries for each group of 63 containers in a given year if the total should exceed 63.



CNTRPT
Start
Are Any Containers Active?
Yes
No
Initialize IFAWD
Print Year Headings
Go to Next Container
Is This Container Active?
No
Yes
Zero Totals
Get Next Leg Name
Search Cargo Table For This Container On This Leg In Up Direction
Print Leg Totals
Is This Container To Be Costed?
Yes
Add Containers To IFA Table
No
Accumulate Leg Totals Into TOT
Finished Searching Cargo Table?
No
Yes
Print Grand Totals TOT
Yes
Finished With All Containers?
No
Return

SECTION 4

SUBROUTINE COLECT

COLECT is called by LEGPRO to shorten the NFTBL table. It is called after a call is made to TRAFIC or when such a call might be made. The call is also made when the NFTBL table is in danger of overflowing subject to the restrictions: 1) the leg being processed must be the terminal leg of a chain (next leg is NONE) or 2) the CALVEH flag is zero. Otherwise an error message is printed and the flights for that leg are not entered into the table.

COLECT splits the NFTBL table into a two-dimensional array, MTT. MTT (I, J) is partially flights of reusable vehicles and partially flights of expended vehicles. I is the vehicle number; J is the relative year number; the NFTBL array is collected from MTT array for like vehicle - year combinations.

SECTION 5
SUBROUTINE CSTRPT

CSTRPT prints the cost reports for facilities acquisitions and vehicle production, development and operations in each year. This is an optional report which may be requested by input of a REPORT COST card to the report table. If the short cost report has been requested, only total costs and program costs are printed; otherwise, costs are itemized for each vehicle, facility and mission. CSTRPT is called by subroutine REPORT.

Costs are accumulated and totalled in 6 arrays (COST1, COST2, COST3, COST4, COST5, COST6) which are equivalenced to consecutive portions of the scratch array DVTT. Each array consists of 33 words, the first two being a cost name; the third the total cost for all years; and the remaining words, the total costs for each year from 1970-1999. Array VDATA, also 33 cells, has the same format except that the entries are annual counts or numbers of units rather than costs. Array LINE is the same as VDATA and equivalent to it in core except that counts are sometimes in floating point and sometimes in integer format.

CSTRPT is executed twice. On the first pass (JFLAG = 1), costs are computed but not printed. At the end of the first pass, the routine examines the total cost array COST6 to find the first year and last year in which the total cost is nonzero. Costs will be printed only for the years between these two limits (inclusive). Printing takes place on the second pass (JFLAG = 0).

VEHICLE DEVELOPMENT AND PRODUCTION COSTS

The first section of the routine computes and prints the vehicle production and development costs. First, array LINE is set to contain the years 1970-1999 (last two digits only), to be used for printing. Then, for each vehicle, CSTRPT

1. Determines how many units of that vehicle were acquired in each year, the information being stored in array LINE;
2. Converts the unit counts from integer to floating point by transferring the data from LINE to VDATA;

3. Extracts from the vehicle table the parameters necessary to find the values of development and production costs for this vehicle and the pointers to the spreading functions to be used;

4. Calls SPDAP to compute the spread development costs for each year in array COST1 and spread production costs in array COST2; Sums both cost types in array COST3 and also accumulates total costs for all vehicles in array COST6;

5. Calls subroutine DPAGER to print the costs for 1970-1984 and save these for 1985-1999 on tape. When all vehicles are finished, DPAGER is called again to print the 1985-1999 costs.

FACILITY ACQUISITION COSTS

The next section of the code generates the IFAC array. Each facility mentioned in the facility acquisition table (IFA) generates a 3-word entry in the IFAC array:

Word 1 - index (NCEP) of this facility in the cargo element table;

Word 2 - year (NYR) that the facility was first used;

Word 3 - program/mission (NPM) which first used the facility (development costs will be charged to this program/mission);

The facility acquisition cost report is generated in the long DO-loop which constitutes the third block of code during the branch LX = 1 (LX = 2 is for the vehicle operations cost report). Most of this DO-loop is executed for both branches, with two exceptions:

1. Facility procurement costs only (LX = 1): block of code between statement #370 and # 480.

2. Vehicle operations costs only (LX = 2): 6 statement lines ending at statement #350.

At the beginning of the loop, the code creates and prints the proper headings and dates, zeroes out the cost arrays, and initializes some key variables. IFLG is a print and flow control flag which assumes the value 3, 2 or 1 depending on whether the routine is about to process a new program, same program but new mission, or same program and mission but new vehicle. IFLG is initially set to 3 and altered by subroutine VEHLDF as it processes parts of the cargo table. NV is not used for facilities. NB is the location of the beginning of a block of cargo items in the phase II cargo table for the current program, mission and vehicle, while NL will later denote the end location of the block. NB is set here to the beginning of the cargo table. CSTRPT uses the cargo table mainly because it is already sorted by program and mission numbers, as are the IVA, IFA and IFAC tables, cutting down on the amount of searching required.

At statement #230, the program number IPRO from the current cargo item block is extracted. Accumulated costs from the previous mission and program, if any, are printed by subroutine DPAGER and added into the array COST6 containing total costs for all programs. At statement #260, the mission number IMIS for the cargo block is extracted, and costs for the last mission, if any, are printed. Later on, this point will be returned to whenever the mission number but not the program number changes. IPM is computed as a packed word indicating combined program and mission numbers, 6 bits each, used later for comparison. At statement #280 the vehicle number IVEH is extracted (but not used for facilities costs) following which a branch on IFLG determines whether new program or mission names must be printed. CSTRPT interrogates IFLAG (6), preset in RDRPT to 1 or 2 to indicate a long or short cost report; if a short report is desired, costs are itemized by program only, not by individual missions.

Statement #340 calls subroutine VEHLDF, which computes load factors for this program/mission/vehicle, changes IFLG, and sets the variable NL pointing to the end of the current cargo block in DDB (compare with variable NB). The load factors are not of interest for facility costs, but the new value of IFLG and NL are. For the facilities report, the routine is also not interested in individual vehicles, just in programs and missions; thus, if IFLG = 1, the routine merely resets NB to the beginning of the next cargo block and cycles back to VEHLDF (provided the cargo table has not been completely examined).

As soon as the program or mission number changes in the cargo table, the test following statement #340 routes the computations to statement #370, where the facilities costs are computed. Statement #370 initializes JF, which points to the current position in the facility acquisition table IFA, which has been presorted by program, year, and facility index.

At #380, subroutine FACNUM is called to compute and store in array LINE the number of units acquired in each year for that program, mission, and facility which belong to the block of entries starting at position JF in IFA. JF is then reset by FACNUM to the beginning of the block of entries for the next program/mission/facility combination stored in IFA. Upon return from FACNUM, CSTRPT determines whether the program/mission combination NPM for the data in array LINE is the same as the desired combination IPM. If not, the program just recycles back through FACNUM until the facility acquisition table is exhausted (JF > NIFA) or data for the current program/mission is found.

When the data is found, the routine extracts the cargo element index N for this facility, the beginning location ICE of data in the cargo element table, the facility index IFAT, and the beginning location M of data in the facility table. The DO-loop ending at statement #400 examines the IFAC table to see if this program/mission was the first to purchase this facility; if so, this program/mission is charged for the facility development costs. Subroutine SPDAP is called to compute the costs for each year according to the development spread function for this facility. DPAGER prints the data. The program/mission number for this entry in the IFAC table is zeroed out so that the routine will skip this entry on all subsequent passes. Following statement #450, SPDAP is called again to compute recurring production costs for each year for this facility, program and mission, based on the production spread function and on the count of units purchased in each year, which is still in array LINE. Again, DPAGER prints the costs. The routine now cycles back to statement #380 for the next program/mission/facility.

From statement #490 to #540, CSTRPT merely accumulates and prints totals and grand totals. At the very end, DPAGER is called with the code word "COPY" to print the costs for 1985-1999 (see DPAGER writeup).

VEHICLE OPERATIONS COSTS

This report is obtained during the DO-loop branch LX = 2. The procedure for printing and working through the cargo table is the same as that for the facility cost report. The actual costs are computed in a small set of 6 FORTRAN statements ending at #350.

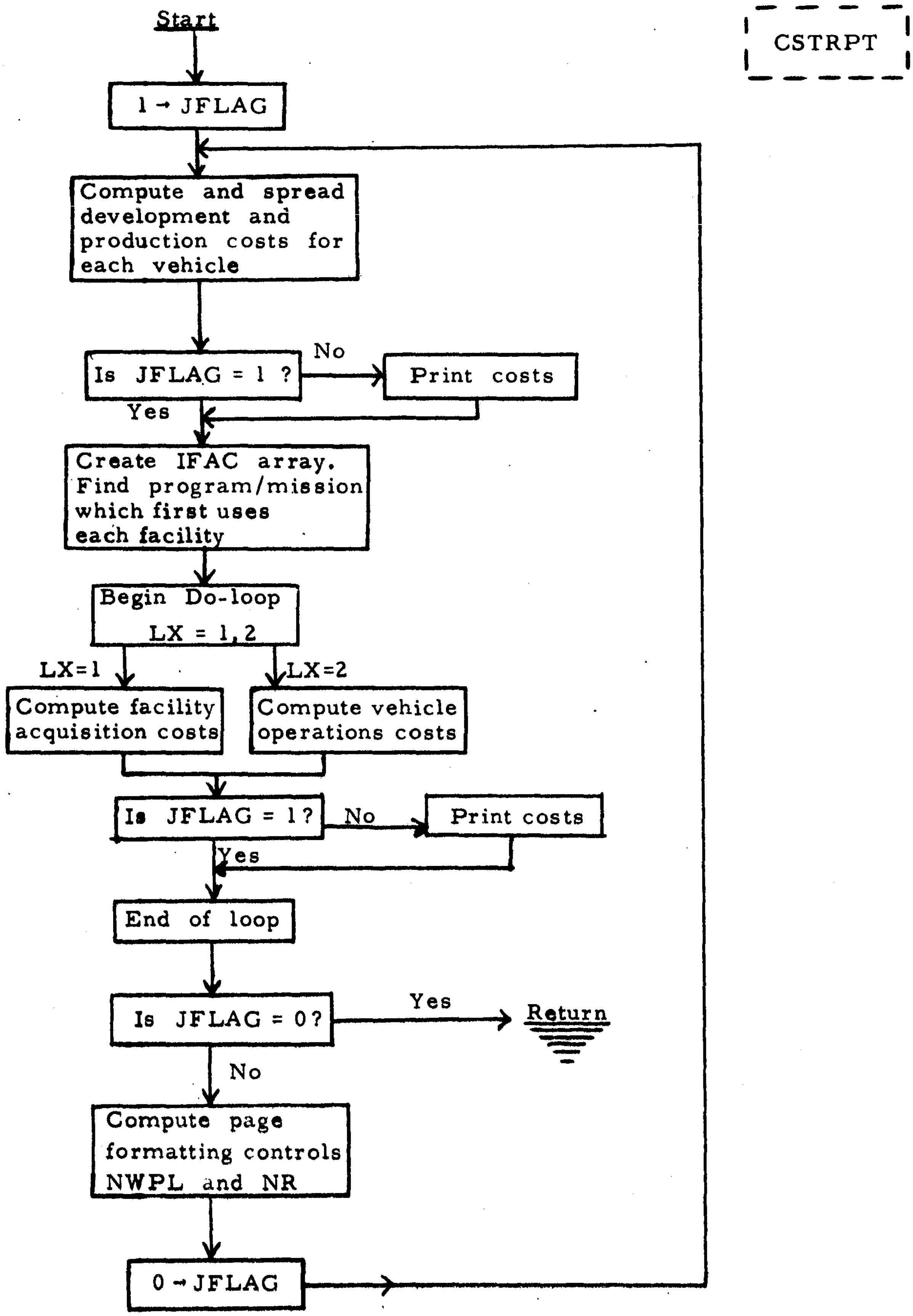
At statement #340, VEHLDF computes and stores in VDATA the sums of load factors for the current program/mission/vehicle in each year. Each sum actually represents some number of whole round-trip flights plus a fraction of a flight for a given year. Hence the operations costs for this year will be computed in SPDAP by multiplying the number of flights by the operations costs for one flight. Variable NV, extracted from TBVEH, is set to the beginning of data for vehicle #IVEH in the DDB array. NV + 13 is the location of operations costs for this vehicle. Since operations costs are to be paid in the year incurred with no spreading, SPDAP is called with the value zero passed as the spread function as a signal indicating no spreading.

PAGE FORMATS

The last block of code is executed on the first pass through CSTRPT. The page formats are determined. If KFLAG = 1 (obtained from IFLAG(11) which is set via input), the page size will be 8 1/2 x 14. NWPL is the number of costs per line of print - either 8 (for 8 1/2 x 11 pages) or 12 (for 8 1/2 14 pages); i.e., NWPL is the number of years covered by each page.

NFW and NLW are the indexes in array COST6 corresponding to the first and last years in which total costs are non-zero. Only these years, a span of NTW years, will be printed. Based on NTW and NWPL, the routine determines how many pages (NR) will be necessary to print costs for all NTW years. NR $\leq$ 4. MWPL is the number of words per line on the last page, if it is not full; MWPL $\leq$ NWPL.

During the second pass through CSTRPT (JFLAG = 0), costs for each group of NWPL years are written on separate scratch tapes and ultimately on separate pages. Printing is done in CSTRPT and in subroutine DPAGER.


CSTRPT
Start
1 → JFLAG
Compute and spread development and production costs for each vehicle
Is JFLAG = 1 ?
No
Print costs
Yes
Create IFAC array. Find program/mission which first uses each facility
Begin Do-loop LX = 1, 2
LX=1
LX=2
Compute facility acquisition costs
Compute vehicle operations costs
Is JFLAG = 1 ?
No
Print costs
Yes
End of loop
Is JFLAG = 0 ?
Yes
Return
No
Compute page formatting controls NWPL and NR
0 → JFLAG

SECTION 6

SUBROUTINE DDBSFT

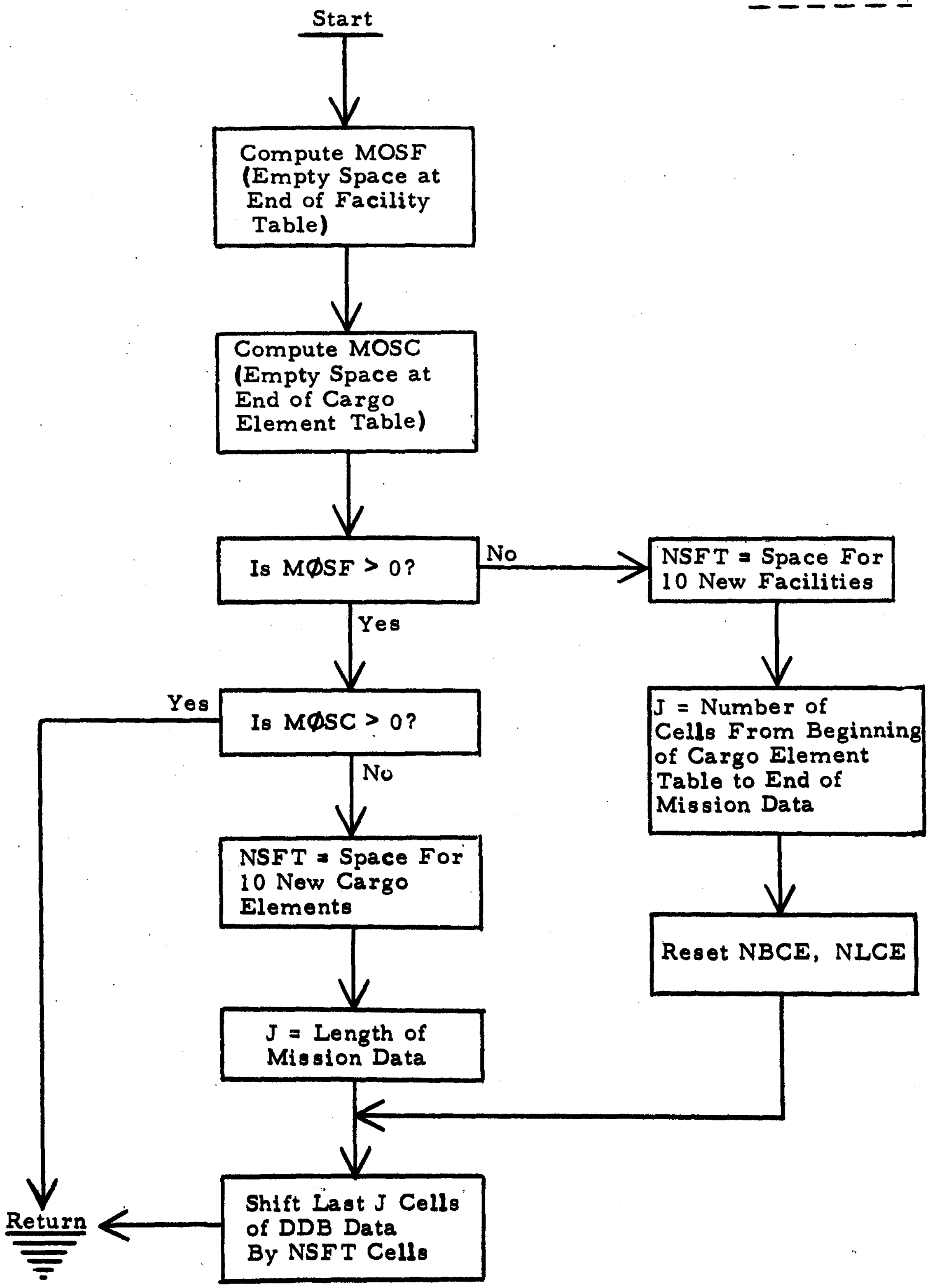
DDBSFT is part of the code that enables the user to interrupt the input of mission data at any times in order to input more facility and/or cargo element data.

The facility table,cargo element table, and Level I cargo table (mission data) are contiguous in array DDB. Thus, to add new facility data once the program has finished defining the location of the cargo element table and the beginning of the Level I cargo table, the program must shift the cargo element and mission data to provide space for the new insertion. Similarly, DORCA must shift the mission data to insert new cargo element data.

To handle this problem, DORCA initially provides enough empty space at the end of the facility and cargo element tables to insert 10 new facilities NOSF cells and 10 new cargo elements (NOSF cells). DDBSFT is called from RDMISS whenever a facility or cargo element is to be inserted. DDBSFT checks the space left at the end of the facility table (MOSF cells) and the cargo element (MOSC cells). If either is zero, the rest of the DDB array is shifted enough to add 10 new entries. Pointers to the beginning and end of the cargo element table (NBCE and NLCE) and Level I cargo table (NBMISS and NLMISS) are adjusted as necessary when a shift is made.

DDBSFT is also called from LEGPRO to add certain program-created items to the cargo element table: boxes representing ground-based, non-longshoring, or propellant off-loading operations.

DDBSFT



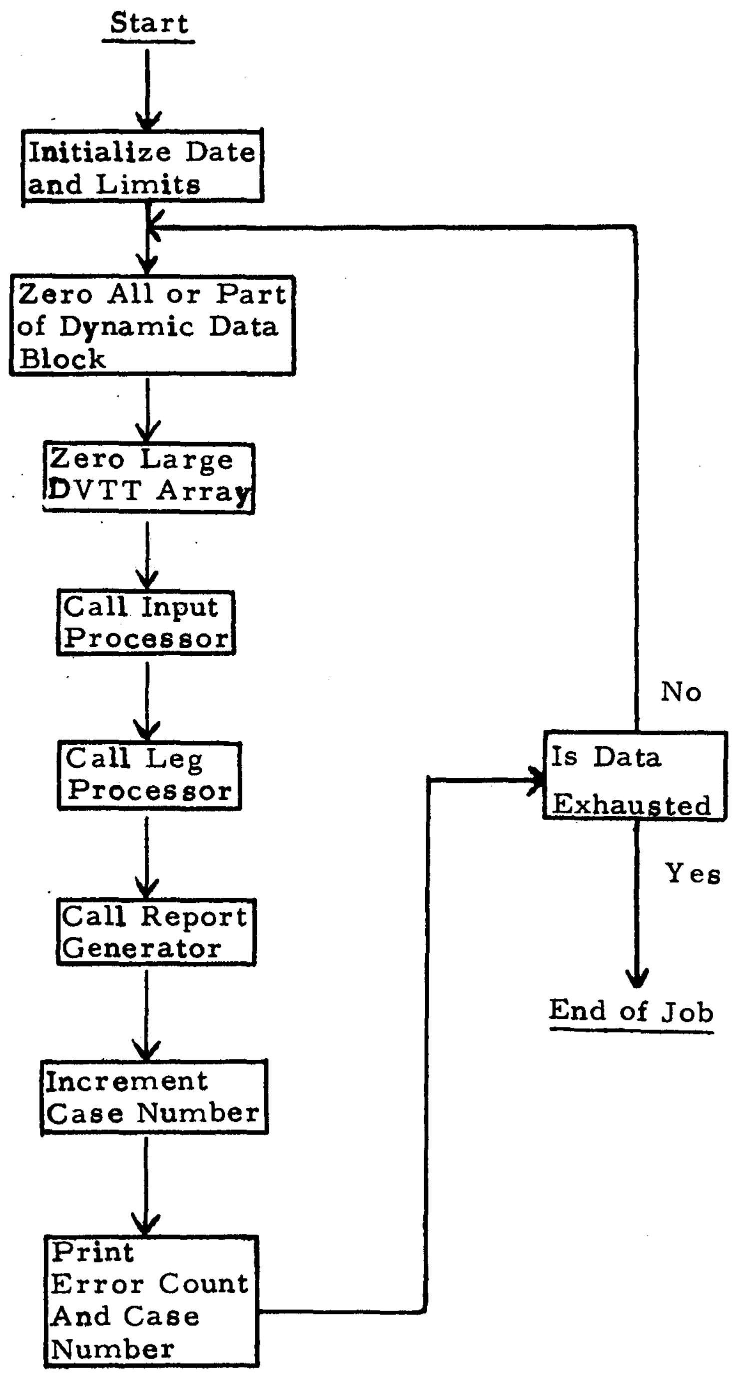
Start
Compute MOSF (Empty Space at End of Facility Table)
Compute MOSC (Empty Space at End of Cargo Element Table)
Is MØSF > 0?
No
NSFT = Space For 10 New Facilities
Yes
J = Number of Cells From Beginning of Cargo Element Table to End of Mission Data
Yes
Is MØSC > 0?
No
NSFT = Space For 10 New Cargo Elements
Reset NBCE, NLCE
J = Length of Mission Data
Shift Last J Cells of DDB Data By NSFT Cells
Return

SECTION 7
ROUTINE DORCA

DORCA is the main routine which controls the overall program flow.

DORCA initializes certain variables and then zeroes out the DDB array between subscripts NBMISS and LOWCOR. LOWCOR is set to the last cell of DDB. After subroutine RDMISS is called, NBMISS indicates the beginning of mission data in DDB, but prior to the first case, NBMISS = 1, so that the entire DDB array is set to zero. On the second and subsequent cases, NBMISS will contain some value substantially greater than 1, so that the zeroing process leaves intact all input data (spread functions, vehicles, facilities, costs and cargo elements) which are stored in DDB preceding the mission data.

DORCA next calls the input processer INPRO, the leg processer LEGPRO and the report generator REPORT. After the case is finished, DORCA prints the case number, the number of fatal input errors, the number of items in the phase I and II cargo tables, and the "gap" or number of unused cells in DDB between the phase I and II cargo tables. Control then passes to the code which zeroes DDB in preparation for the next case.


Start
Initialize Date and Limits
Zero All or Part of Dynamic Data Block
Zero Large DVTT Array
Call Input Processor
Call Leg Processor
Call Report Generator
Increment Case Number
Print Error Count And Case Number
Is Data Exhausted
No
Yes
End of Job

SECTION 8

SUBROUTINE DPAGER

DPAGER is called by CSTRPT to help format and print the various cost reports.

The format is arranged so that each page of printout contains costs for a span of NWPL years (NWPL = 8 or 12) and NR pages are required to print costs for all the years ($1 \leq NR \leq 4$). The variables NWPL and NR are computed by CSTRPT, which chooses the time span so that any years at the beginning and/or end of the study period 1970-99 in which no costs are incurred are omitted from the printout.

Costs are forwarded to DPAGER in array LINE of 33 cells. The first two words of LINE are the cost name, word 3 is the total cost for all years, and the remaining 30 words are the annual costs for each of the 30 years from 1970-99. Variables NFW and NLW are indices referring to array LINE and corresponding to the first and last years of nonzero costs.

The printing process consists of a storage phase followed by a copy phase.

Storage phase. DPAGER writes the cost name [LINE(1-2)] plus the first group of NWPL costs (starting at index NFW) onto scratch tape #1; the cost name plus the next NWPL costs onto scratch tape 2; the cost name plus the next NWPL costs onto tape 3 if necessary; and the cost name plus the last NWPL costs onto tape 4 if necessary. This action does not take place if any of the following conditions are true:

JFLAG = 1 (indicates CSTRPT is in its first pass and no printing is required at this time).

LINE(3) = 0 (indicates that no costs were incurred in any year for this cost name).

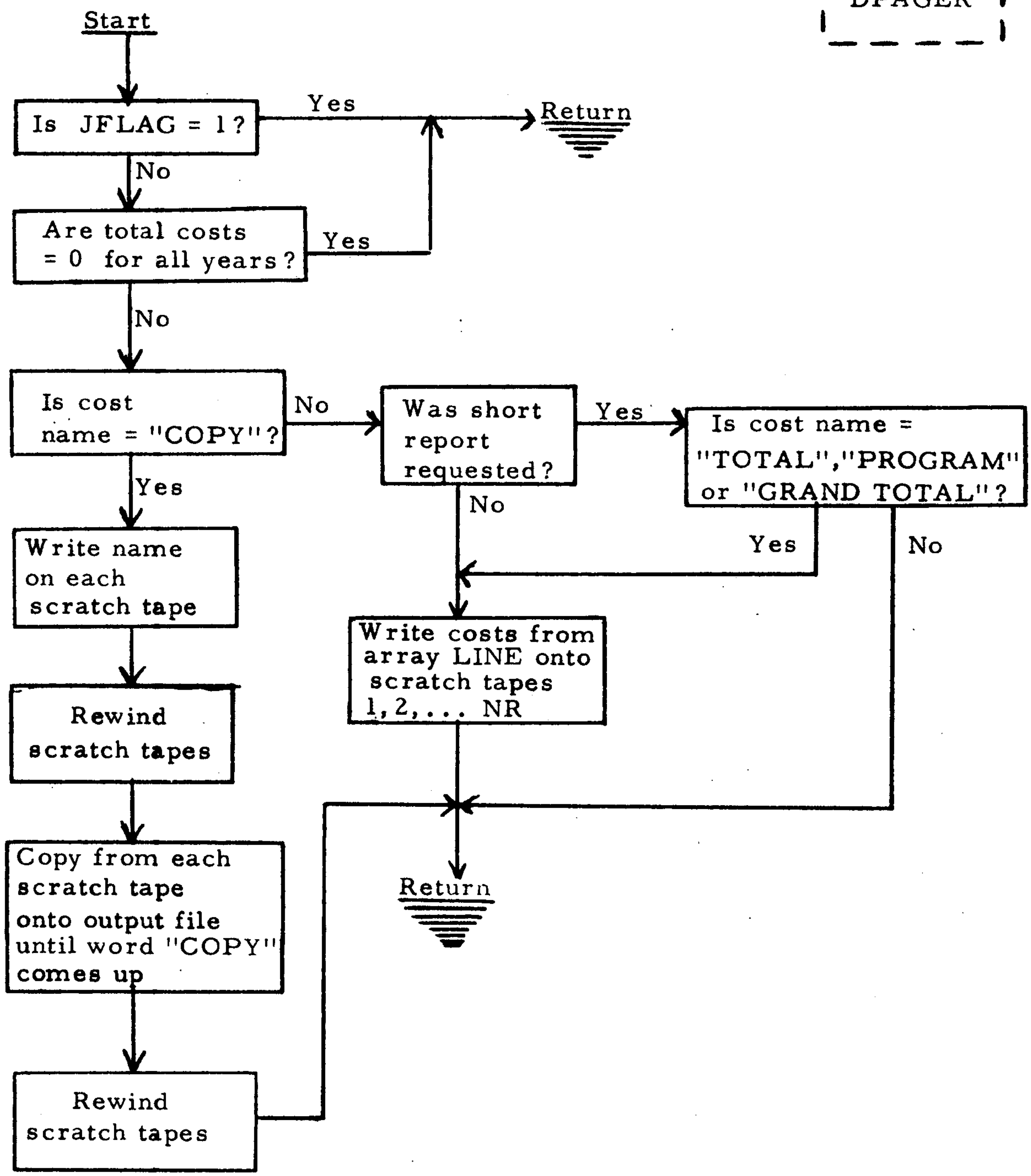
LINE(1) = "COPY" (indicates the copy phase is required, not the storage phase).

In addition, the routine must check the value of IFLAG(6), according to the following table:

$$\text{IFLAG(6)} = \begin{cases} 0 & \text{for no cost report} \\ 1 & \text{for the standard (long) cost report} \\ 2 & \text{for the short cost report.} \end{cases}$$

DPAGER is not called if IFLAG(6) = 0. If the short report was requested DPAGER prints and saves only when the cost name is one of the three key words "TOTAL", "GRAND TOTAL", or "PROGRAM".

Copy phase. If the cost name in LINE(1) is found to be the key word "COPY", it is taken as a signal to print the cost data saved on previous entries to DPAGER since the last copy operation. The key word "COPY" is written onto each of the NR scratch tapes being used and the tapes are rewound. Data is copied from tape #1 onto the print file, using as many pages as necessary, until the key word "COPY" is encountered again. Next, data is transferred from tape 2 to the print file, then from tapes 3 and 4 if necessary. Then the scratch tapes are rewound and DPAGER exits.


Start
Is JFLAG = 1?
Yes
Return
No
Are total costs = 0 for all years?
Yes
No
Is cost name = "COPY"?
No
Was short report requested?
Yes
Is cost name = "TOTAL","PROGRAM" or "GRAND TOTAL"?
Yes
Write name on each scratch tape
No
Yes
No
Write costs from array LINE onto scratch tapes 1, 2, . . . NR
Rewind scratch tapes
Copy from each scratch tape onto output file until word "COPY" comes up
Return
Rewind scratch tapes

SECTION 9
SUBROUTINE FACNUM

FACNUM counts the number of units of a given facility to be acquired in each year for a specified program and mission.

FACNUM is called by subroutines FACRPT and CSTRPT with two arguments: an array A of 33 cells to be filled with the yearly totals, and an index JF which is a pointer to the facilities acquisition table IFA. The facilities acquisition table, which is described in section III. B. 4, has been presorted so that entries are in numerical order and all entries referring to the same program, mission and cargo element (ie, facility) are contiguous.

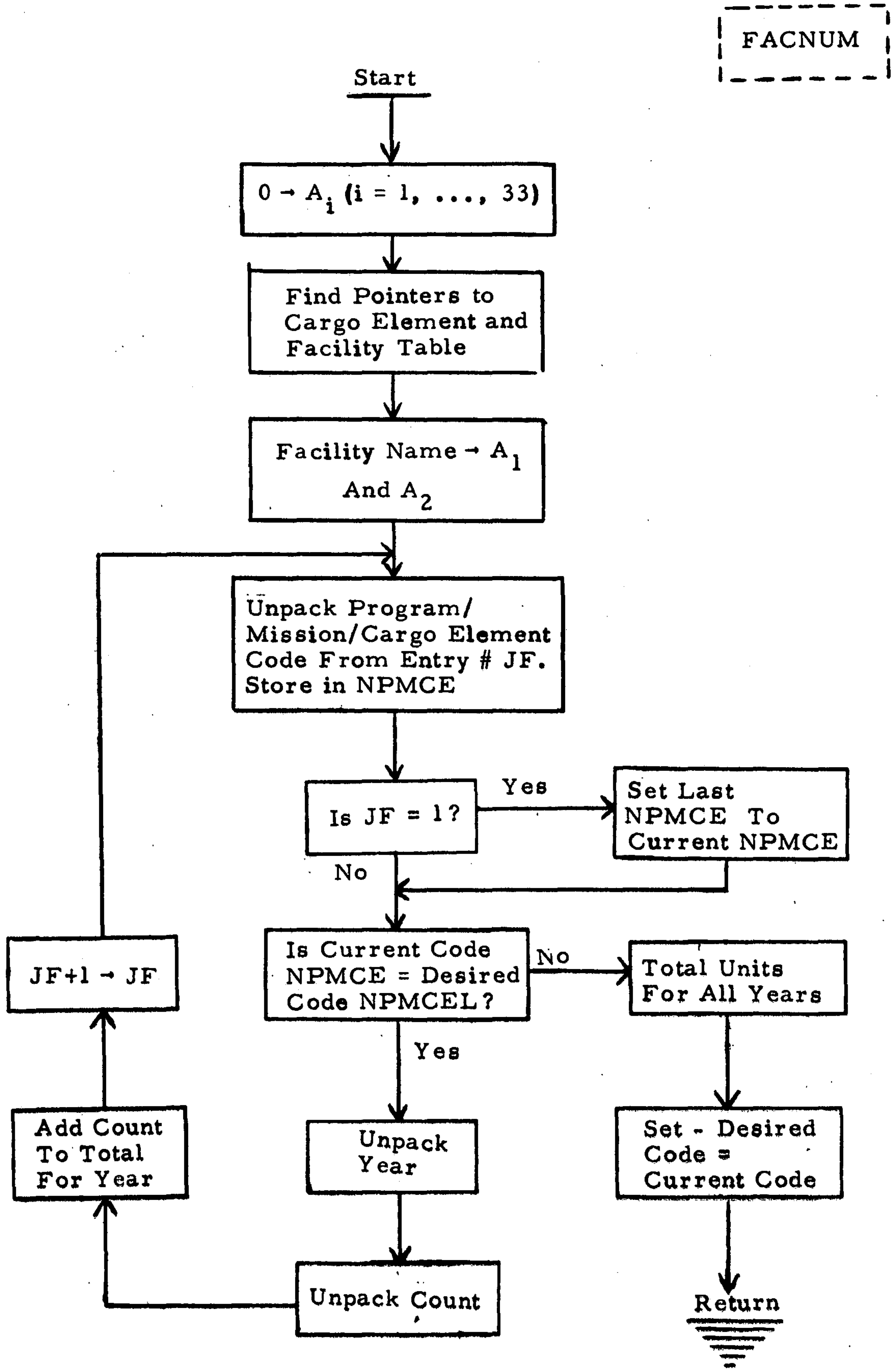
The pointer JF is preset in FACRPT and CSTRPT to the first entry in IFA for the program, mission and cargo element currently being processed by FACRPT. From this first entry, IFA(JF), the routine extracts the program/mission/cargo element code of 24 bits and stores it in variable NPMCEL. Also from this first entry the routine finds the index N of this facility in the cargo element table, from N it computes the pointer ICE to the exact cell in DDB which it needs, and from ICE it computes another pointer M to the facility table in DDB. DDB(M) and DDB(M + 1) contain the facility name, which are now transferred to array A (initially zero).

Now, repeatedly incrementing JF by 1, the routine searches through the acquisition table IFA. FACNUM extracts the program/mission/cargo element code NPMCE from each entry and compares it with the desired code in NPMCEL. If the two quantities match, then the year (relative to a starting date) and count are extracted and the count is added to the appropriate cell of array A. A has the following format:

Word	Contents
1 - 2	Facility name (A6, A4 format)

Word	Contents
3	Total for all years
4	Total units acquired in year 1
5	Total units acquired in year 2
6	Total units acquired in year 3
. . .	etc.

The first time NPMCE $\neq$ NPMCEL, the search is terminated. JF is left at this value, which will be the first entry for the next program/mission/cargo element. The sum of all units acquired in all years is now entered in A (3). NPMCEL is reset to the new code still residing in NPMCE, ready for the next call to FACNUM.


FACNUM
Start
0 → A_i (i = 1, ..., 33)
Find Pointers to Cargo Element and Facility Table
Facility Name → A_1 And A_2
Unpack Program/ Mission/Cargo Element Code From Entry # JF. Store in NPMCE
Is JF = 1?
Yes
Set Last NPMCE To Current NPMCE
No
Is Current Code NPMCE = Desired Code NPMCEL?
No
Total Units For All Years
Yes
Unpack Year
Set - Desired Code = Current Code
Unpack Count
Return
Add Count To Total For Year
JF+1 → JF

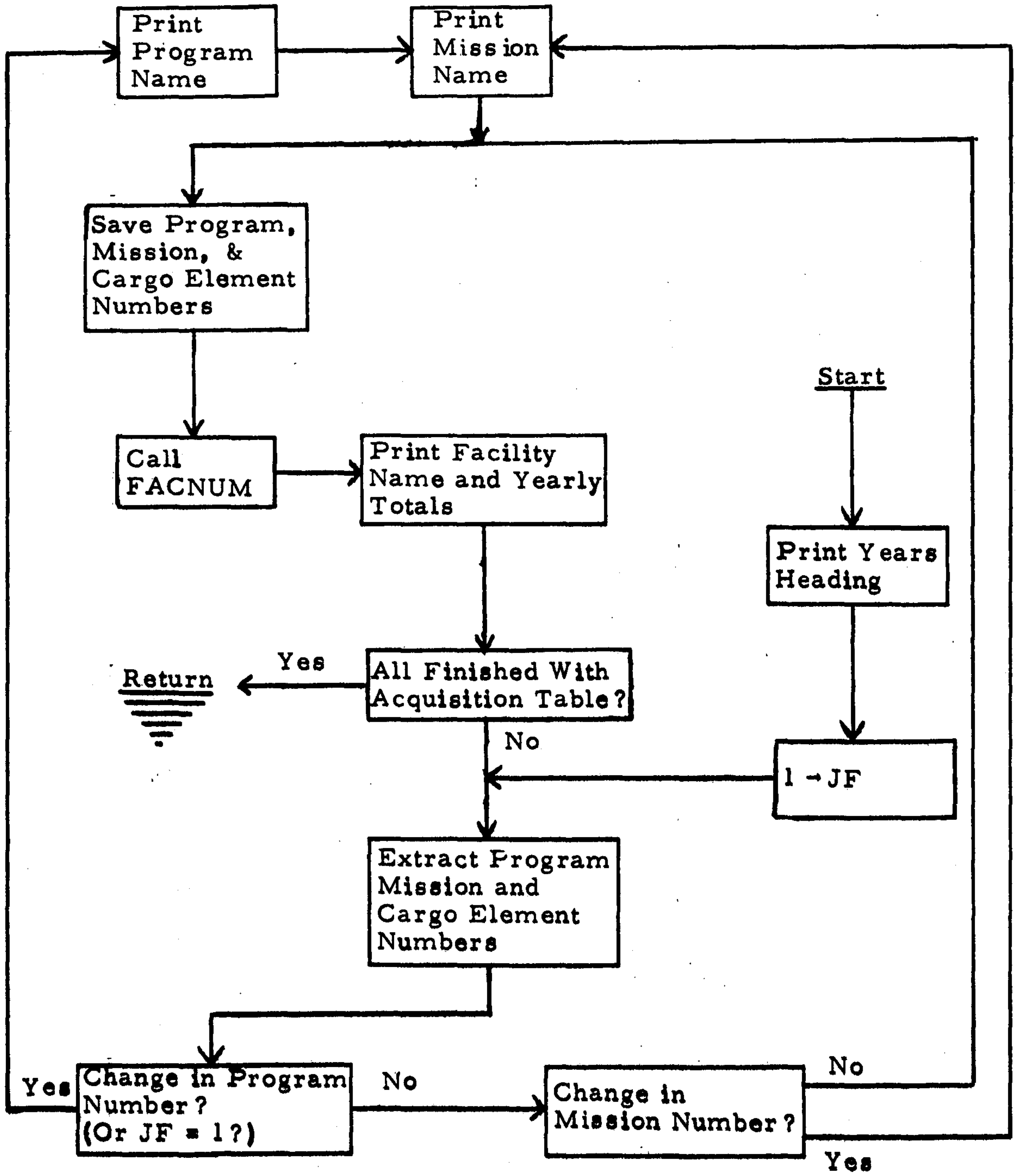
SECTION 10
SUBROUTINE FACRPT

FACRPT produces the facility schedule report if it has been requested by input. The facility acquisition schedule lists the total units of a given facility acquired in each year, listed by program and mission.

FACRPT is a very simple routine which is called by subroutine REPORT. FACRPT call subroutine FACNUM to process small portions of the facility acquisition table IFA (see section III. B. 4) and store the totals in array A. FACRPT prints the totals from array A, changing the program heading or mission subheading whenever a change occurs in the program number NP or mission number NM in any entry in the IFA table.

See FACNUM for the format of array A.

FACRPT


Print Program Name
Print Mission Name
Save Program, Mission, & Cargo Element Numbers
Call FACNUM
Print Facility Name and Yearly Totals
Start
Print Years Heading
Return
Yes
All Finished With Acquisition Table?
No
1 → JF
Extract Program Mission and Cargo Element Numbers
Yes
Change in Program Number? (Or JF = 1?)
No
Change in Mission Number?
No
Yes

SECTION 11

SUBROUTINE FIND

FIND is called by ASINER to search the list of remaining unassigned cargo items (WS matrix) for one satisfying the specified MODE, TYPE, DIRECTION, maximum weight, volume and round-trip requirements, and deployment limits. If the type is bulk, the kind must also match the kind specified by ASINER.

If more than one item satisfies all requirements, the largest one is chosen. If this assignment implies adding a new crew capsule or bulk container, then the combined weight of cargo item and container must not exceed the specified weight limit (computed in terms of equivalent up-weight). If the type is bulk and no item smaller than the weight limit exists, the program will choose a larger one satisfying all other requirements and assign only that portion which will fit, leaving the remainder as an entry in the unassigned cargo list.

The success or failure of the search is indicated upon return from FIND by the common variable INDEX. If the search was a failure, INDEX = 0. If the search was successful, INDEX will be set to a positive integer, which represents the subscript in the WS array where the item was found. If the search is successful in that a bulk item was split up, INDEX will be set to the negative value of the subscript of the item in WS. The zero or nonzero state of INDEX is used as a signal to ASINER that the search was or was not successful, and the positive or negative state is used subsequently by FIND.

The weight limit WTLIM not to be exceeded is first taken as VCAP, remaining vehicle capacity. If the direction specified is down, WTLIM is divided by the factor UPMAX/DNMAX, which reduces WTLIM. If the assignment implies adding a new bulk container, the capacity TCAP and empty weight CW of that kind of container must be considered. The true upper limit WTLIM of the cargo weight which can be assigned at this point on this flight is given by
WTLIM = MIN {WTLIM - CW, TCAP}

Once WTLIM is determined, the routine examines each item remaining unassigned in the WS list. Acceptable candidates must satisfy the following requirements.

(a) Move, type, direction equal to those specified.

(b) Weight limits

For crews, weight W + capsule weight $\leq$ WTLIM.

For discrete items, W $\leq$ WTLIM.

For bulk, no limit since it can be subdivided.

(c) For bulk, container index ICT must match the required index MCT.

(d) Volume limits

The volume of the item (if discrete) or its capsule/container (if crew or bulk) must not exceed remaining vehicle volume RVOL in this direction.

(e) Deployment limits. If this item requires single deployment, the number of items previously assigned to this flight in this direction must be zero.

(f) Round-trip requirements. If item must travel round trip on this flight, weight, volume, and deployment limits must be satisfied in opposite direction also.

Of all acceptable candidates, the heaviest one will be chosen for assignment.

If the search is successful, the actual assignment procedure consists of several steps:

1. If the assignment implied adding a new container, FIND must

   (a) Reset container capacity CCAP.

   (b) Create an entry for this container in the CR array (container requisition list).

   (c) Subtract container empty weight (effective up weight) from remaining vehicle capacity VCAP.

   (d) Add the container weight to matrix TW, which specifies the total weight for each flight in each direction.

   (e) If space permits, schedule round trip of the container on this flight; otherwise, create a new entry in the WS (unassigned cargo) matrix for the shipment of the container as an empty discrete item in the opposite direction on a later flight, and add the container weight to the WLEFT matrix. However, if input requires this container to be expended, do neither.

2. If the item represents the return trip of a requisitioned container,

   (a) Insert the flight number into the entry for this container in the CR list.

   (b) Go to step 4.

3. If the item is not a returning requisitioned container,

   (a) Increment NASS (total number of assignments) by one.

   (b) Create the packed word format as described in subroutine ASINER and store it into matrix FLTA.

   (c) Go to step 4.

4. Adjust running sums and variables.

   (a) If INDEX < 0 (indicating a subdivided bulk item), subtract the assigned weight from the item weight and leave the item in the WS array.

   (b) If INDEX > 0 (indicating an item fully assigned), delete it from WS and replace it by the last item in WS. Change ITEMS by one. Decrement NMODE (MODE) by one.

   (c) If this is a discrete item or if a new crew capsule or bulk container was added (FLAG = 1), add one to NOCC (direct), which is the count of cargo items assigned to this flight in this direction.

   (d) If item is crew, set MANNED flag to indicate direction(s) in which crew is travelling.

   (e) Subtract equivalent up weight of this item from VCAP.

   (f) Add actual weight of item to TW matrix.

   (g) If this is bulk cargo (TYPE = 2), subtract the assigned weight from the CD or CD2 matrix and from CCAP (remaining container/capsule capacity).

   (h) If this item is a portion of bulk cargo from the capture bin, set the "mode change" flag MCHG and redefine the remainder as regular (MODE = 3) instead of optional (MODE = 2) cargo.

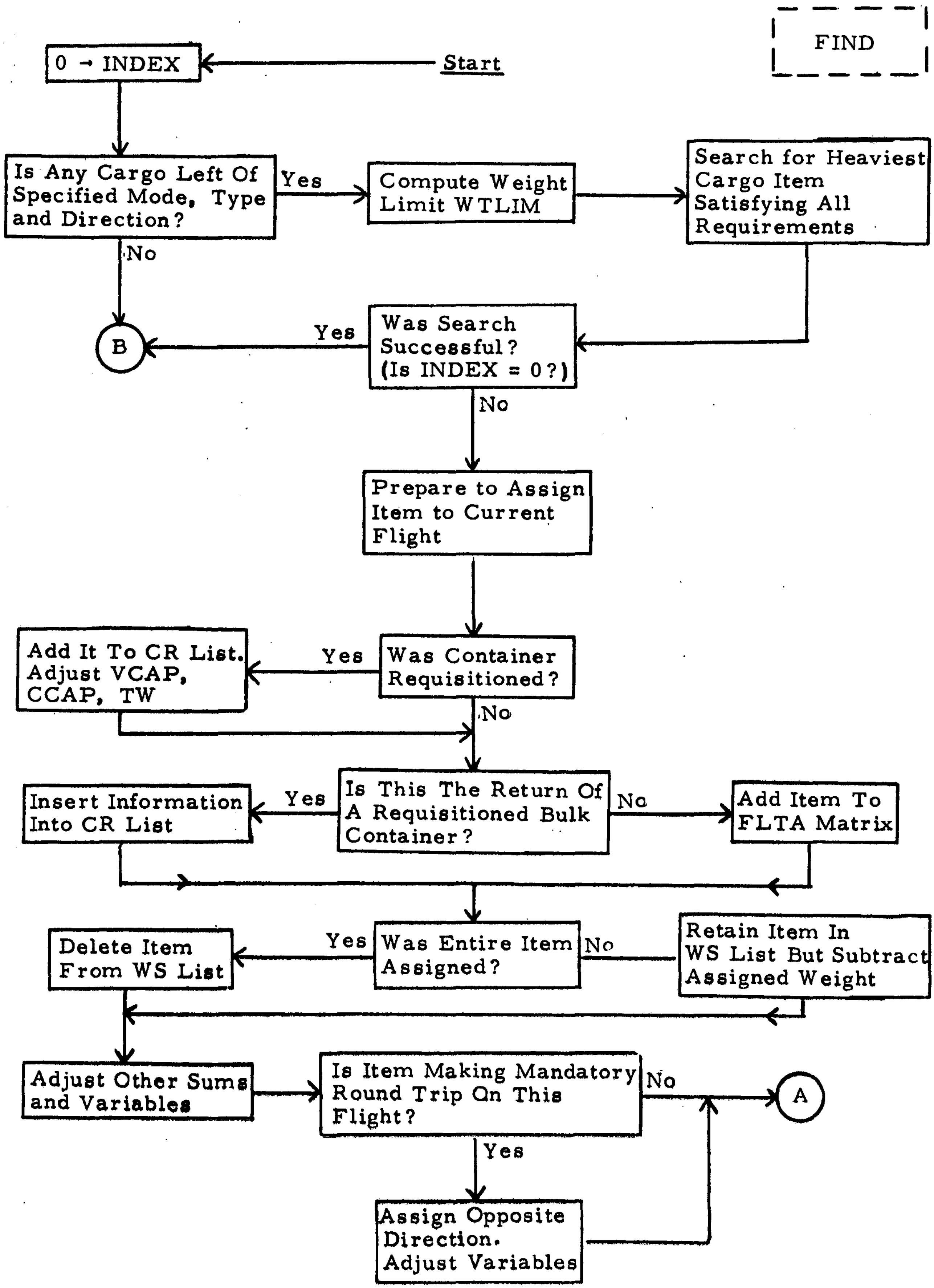
   (i) Subtract the weight of this item from the total weight still unassigned for this mode, type and direction, in the WLEFT matrix.

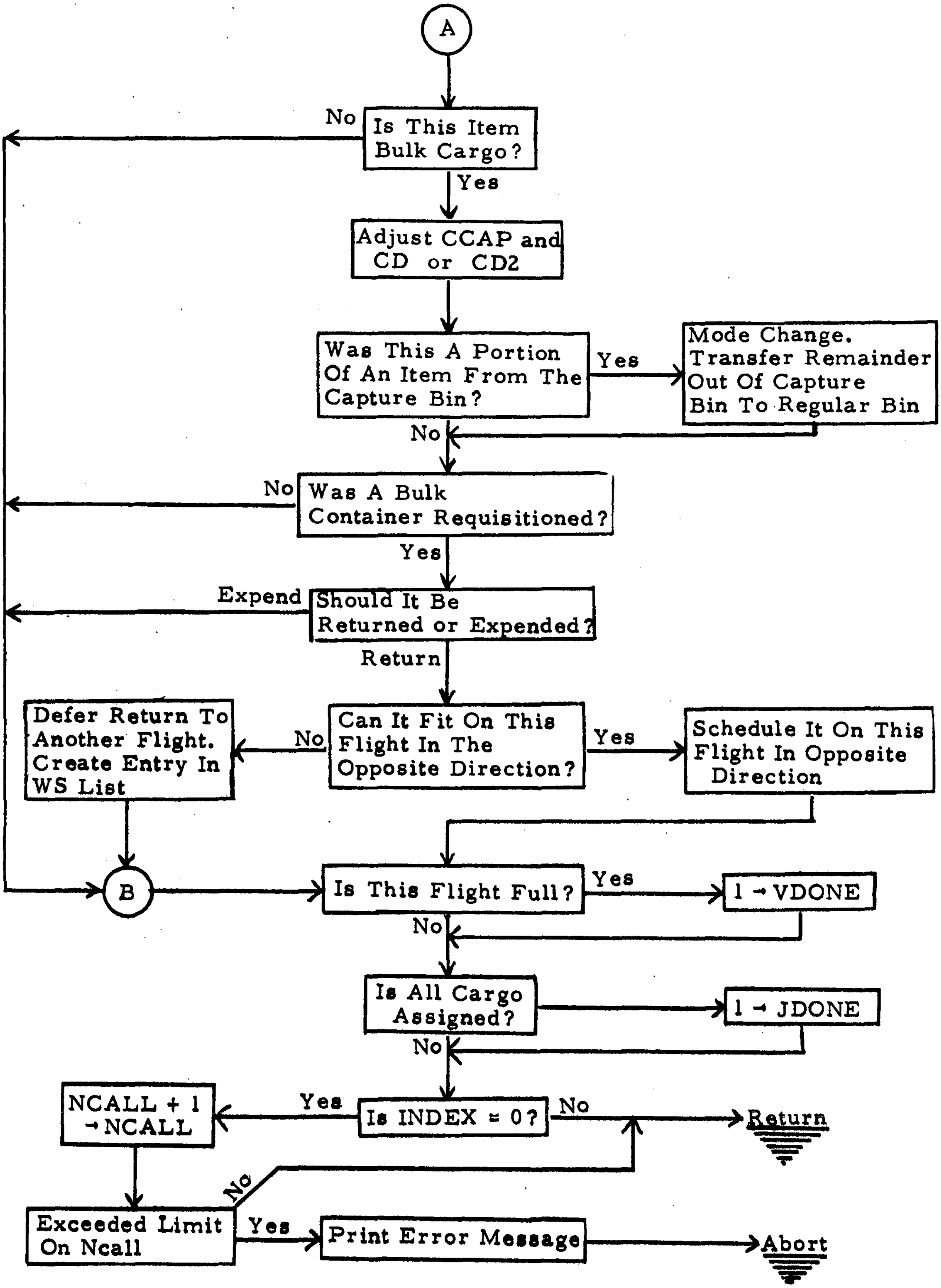
   (j) Subtract from RVOL (DIRECT) the volume of this assignment.

   (k) Adjust variables as necessary for items making round trips on this flight.

To avoid infinite loops in the assignment process due to some unforeseen problem, a counter (NCALL) keeps track of the number of consecutive unsuccessful calls to subroutine FIND. Each time FIND exits with INDEX = 0, NCALL is incremented by 1; if it exceeds its maximum limit LIMCAL (currently 100), the program aborts with an error message and core dump. Each time FIND is successful in its search (INDEX $\neq$ 0) NCALL is reset to zero.

For additional information, including a complete explanation of all coding variables and error printouts, see the writeup for subroutine ASINER.


FIND
0 → INDEX
Start
Is Any Cargo Left Of Specified Mode, Type and Direction?
Yes
Compute Weight Limit WTLIM
Search for Heaviest Cargo Item Satisfying All Requirements
No
B
Yes
Was Search Successful? (Is INDEX = 0?)
No
Prepare to Assign Item to Current Flight
Add It To CR List. Adjust VCAP, CCAP, TW
Yes
Was Container Requisitioned?
No
Insert Information Into CR List
Yes
Is This The Return Of A Requisitioned Bulk Container?
No
Add Item To FLTA Matrix
Delete Item From WS List
Yes
Was Entire Item Assigned?
No
Retain Item In WS List But Subtract Assigned Weight
Adjust Other Sums and Variables
Is Item Making Mandatory Round Trip On This Flight?
No
A
Yes
Assign Opposite Direction. Adjust Variables

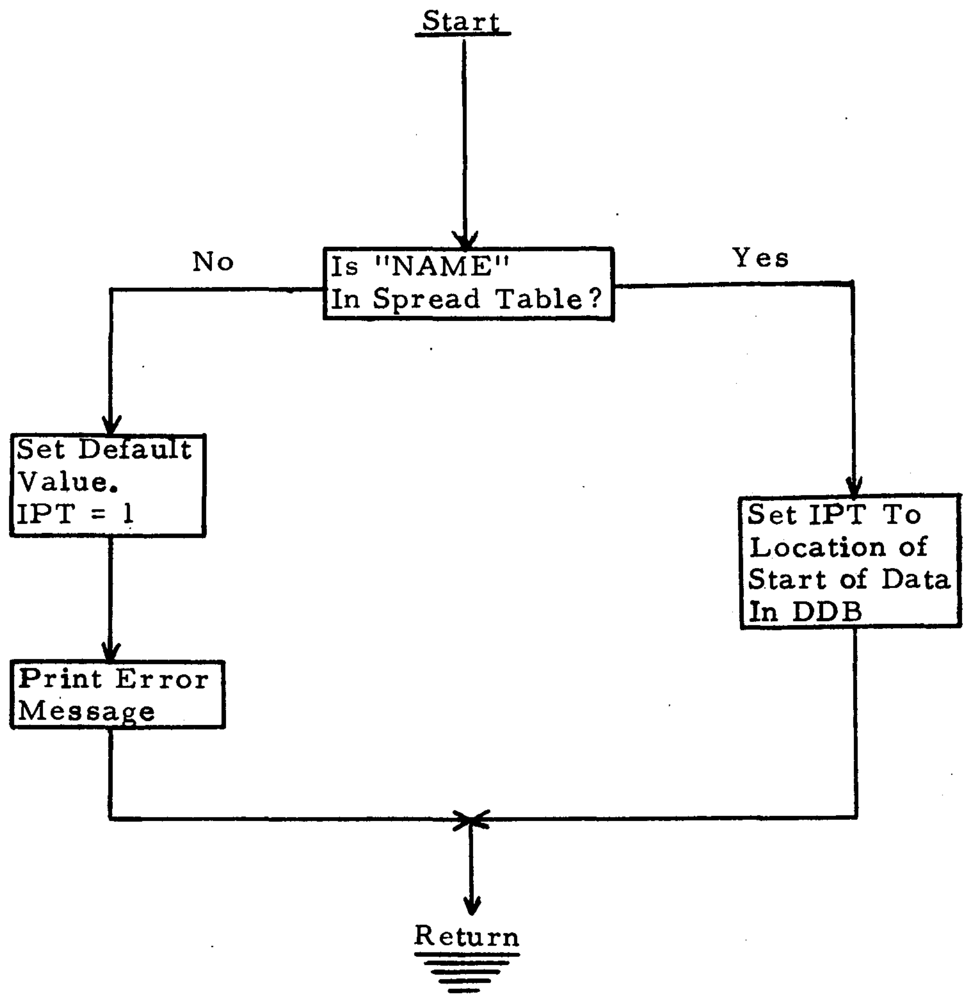

A
No
Is This Item Bulk Cargo?
Yes
Adjust CCAP and CD or CD2
Was This A Portion Of An Item From The Capture Bin?
Yes
Mode Change. Transfer Remainder Out Of Capture Bin To Regular Bin
No
No
Was A Bulk Container Requisitioned?
Yes
Expend
Should It Be Returned or Expended?
Return
Defer Return To Another Flight. Create Entry In WS List
No
Can It Fit On This Flight In The Opposite Direction?
Yes
Schedule It On This Flight In Opposite Direction
B
Is This Flight Full?
Yes
1 → VDONE
No
Is All Cargo Assigned?
1 → JDONE
No
NCALL + 1 → NCALL
Yes
Is INDEX = 0?
No
Return
No
Exceeded Limit On Ncall
Yes
Print Error Message
Abort

SECTION 12
SUBROUTINE FINDSP

FINDSP determines a pointer to the spread table corresponding to a given spread function name.

FINDSP has two arguments: NAME, which is a 2-celled array containing the name of a spread function in (A6, A4) format upon entry; and IPT, which is to be set to the starting location of data for that spread function in the DDB array. Despite the fact that IPT has an integer name, its contents are in floating point. If no entry in the spread table TBSPD matches the contents of NAME, an error message is printed and IPT is set to 1.

FINDSP is called by input routines RDVEH and RDFAC.


Start
Is "NAME" In Spread Table?
No
Yes
Set Default Value. IPT = 1
Print Error Message
Set IPT To Location of Start of Data In DDB
Return

SECTION 13

SUBROUTINE INPRO

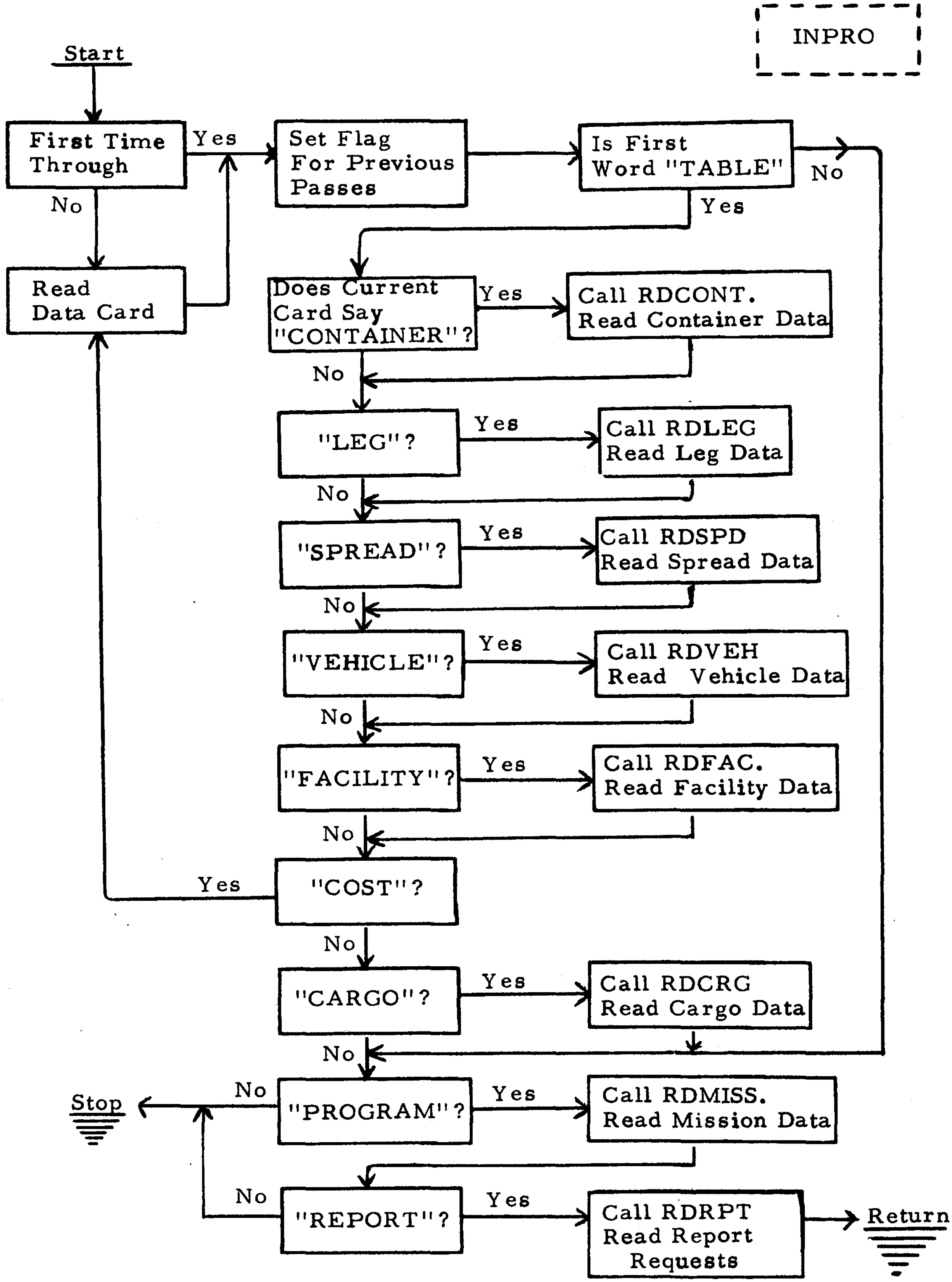
INPRO controls the overall flow of processing input cards.

Actually reading of cards is done by subroutine READER. The first call to READER is from INPRO; all subsequent calls are from the individual input routines. Whenever a card is encountered in which field 1 (columns 1-10) contain the words "TABLE", "PROGRAM", "REPORT" or "$DONE" control is returned to INPRO to call the appropriate subroutine to process the cards.

Tables must be input in the following order:

CONTAINER
LEG
SPREAD
VEHICLE
FACILITY
CARGO
PROGRAM
REPORT

Data out of order will be improperly processed with no error message printed.


INPRO
Start
First Time Through
Yes
No
Set Flag For Previous Passes
Is First Word "TABLE"
No
Yes
Read Data Card
Does Current Card Say "CONTAINER"?
Yes
Call RDCONT. Read Container Data
No
"LEG"?
Yes
Call RDLEG Read Leg Data
No
"SPREAD"?
Yes
Call RDSPD Read Spread Data
No
"VEHICLE"?
Yes
Call RDVEH Read Vehicle Data
No
"FACILITY"?
Yes
Call RDFAC. Read Facility Data
No
Yes
"COST"?
No
"CARGO"?
Yes
Call RDCRG Read Cargo Data
No
Stop
No
"PROGRAM"?
Yes
Call RDMISS. Read Mission Data
No
"REPORT"?
Yes
Call RDRPT Read Report Requests
Return

F

SECTION 14
SUBROUTINE LEGPRO

LEGPRO directs the shipping of cargo so as to satisfy all input mission requirements. The basic task is to schedule all of the cargo specified by input in each year on all of the legs it must travel. The scheduling procedure itself generates additional cargo which must also be scheduled properly: vehicles, propellant used by the carrying vehicles, bulk containers, crew capsules, and boxes for non-longshoring, propellant off-loading, and ground-based satellites.

The cargo specified by input is contained in the phase I (or level I) cargo table, stored in the dynamic data block DDB between the limits of NBMISS and NLMISS in the format described in Appendix B. Each entry represents a single item to be shipped in a single year; multiple entries are created for items of multiplicity greater than one. Each entry represents the cargo item only on its highest leg; shipment of cargo items on all lower legs is accomplished during the processing by modifying the existing entries in the cargo table (not by creating additional ones for lower legs). Additional entries are made to the phase I cargo table to provide containers, vehicles, boxes and propellant tanks when and where needed and to schedule their shipment on lower legs

The actual assignment of individual cargo items is handled by subroutine ASINER. The function of LEGPRO is to organize the input to ASINER and to store on tape the output in the phase II (or level III) cargo table, whose format is described in Appendix B. Each entry in the phase II cargo table represents a scheduled cargo item on a given leg and vehicle in a given year. The item may be one specified by mission input (including a whole or fractional portion of a bulk cargo item), a container requisitioned by ASINER, a vehicle acquired by TRAFIC, a propellant tank requisitioned by LEGPRO to fuel the carrying vehicles, or an off-loaded vehicle. Separate entries are made for each leg on which each item must travel. Thus, we may consider that the phase I cargo table represents only a basic worksheet while the phase II cargo table represents the complete cargo manifest in all details, and that the function of LEGPRO is to process the phase I table into the phase II table.

Subroutine ASINER is called to schedule all cargo items in a given leg/year/vehicle. This means determining the number of flights needed by that vehicle and assigning every item to one of the flights in the proper direction(s). Bulk cargo items are subdivided as necessary to fill up small spaces, and bulk containers are requisitioned as necessary. ASINER assumes that all cargo items for the specified leg/year/vehicle form a contiguous block in DDB between the limits of IF and IL. Initially, the phase I cargo table is sorted by leg and, within each leg, by year and vehicle. After each group of cargo items is selected by LEGPRO and processed by ASINER, the containers requisitioned by ASINER and enough propellant tanks to fuel the flights scheduled are added by LEGPRO to the end of the phase I cargo table for scheduling on lower legs. The algorithm is arranged to process all years and all vehicles for a single leg before going on to the next leg. After all legs originating from a common lower leg are processed, however, the extra propellant tanks and bulk containers added to the phase I cargo table plus cargo coming from or going to higher legs but not connected with the previous common lower leg require LEGPRO to sort the cargo table again so that all items for each leg/year/vehicle form a contiguous block.

The phase II cargo table contains most of the same information as the phase I cargo table, rearranged for easier post-processing, plus three additional pieces of information:

1. The flight number, since normally more than one flight will be necessary to carry all cargo for the given leg/vehicle/year;

2. The load factor, which is a weighted ratio indicating what proportion of the total weight carried by this flight (up and down) represents each cargo item (see Appendix B for the equation defining load factor); and

3. The bulk load factor, which is useful only for bulk cargo items that have been subdivided. This is the ratio of the weight of this portion of the cargo item to its original undivided weight in the cargo element table. For indivisible items such as crews and discretes, the bulk load factor is identically 1. This factor is the only indicator of subdivided bulk cargo.

The phase II data is stored on tape # L2WOT in logical records of 510 words (170 cargo items).

Capture bin

Some cargo, which we call "a priori" cargo, has been designated to fly on a specific vehicle for a given leg and year. On the other hand, "capture" cargo is cargo which is permitted to fly on any vehicle flight which has leftover space which cannot be filled by any regular cargo because of weight, volume, or other constraints.

The procedure is to call ASINER with a list of a priori cargo which must be assigned to a specified vehicle, plus a list of cargo (if any exists) which is available for capture by vehicle on that leg/year. ASINER may assign some, all or none of the capture cargo to flights of the specified vehicle (see writeup on ASINER). Any capture cargo thus assigned is removed from the capture list, and the rest of that list is given to ASINER along with the list of a priori cargo for the next vehicle on that leg/year.

After all a priori cargo for all vehicles on that leg/year has been assigned, some capture cargo may remain unassigned. In this case, the vehicle preference list is used. LEGPRO selects the heaviest remaining capture cargo item and then runs down the vehicle preference until it finds the first vehicle capable of carrying that cargo with a normal (non-expended) return flight; if no vehicle is capable, then LEGPRO selects the first vehicle which can carry the item in expended mode. This heaviest cargo is then redesignated as a priori cargo for the selected vehicle. ASINER then is called with an a priori list of one item and the remainder of the capture list. Hopefully, ASINER will manage to assign some more capture cargo as well as the newly designated a priori cargo. This procedure - selecting the heaviest remaining capture cargo and selecting the first vehicle on the vehicle preference list capable of carrying it - is repeated until the capture list for this leg/year is exhausted.

Propellant requirements (space-based mode)

The flights scheduled by ASINER mean that sufficient propellant must be available at the lower terminus of the leg. In the space-based mode, for each individual flight, knowing the total payload up and down weights carried, LEGPRO determines the amount of propellant required. If propellant off-loading is not permitted, each vehicle is fueled to its maximum (specified in input). If off-loading is permitted , LEGPRO calls PROPCL to compute PROPWT, the actual propellant required for the flight, and PROPOL, the amount off-loaded in the first or lowest stage (all upper stages being filled fo fuel capacity). PROPWT is accumulated in array TPROPW, which contains the total propellant required for all flights of each vehicle on the current leg/year.

LEGPRO now adds to the level I cargo table all the full fuel tanks necessary for all flights of that vehicle on that leg/year; these tanks will become part of the cargo on the next lower leg. In addition, in the case of off-loading, LEGPRO must create a new cargo element with the proper weights and volume corresponding to the first stage of the vehicle plus its off-loaded propellant, to be treated as a unit.

No propellant tanks are shipped in ground-based mode.

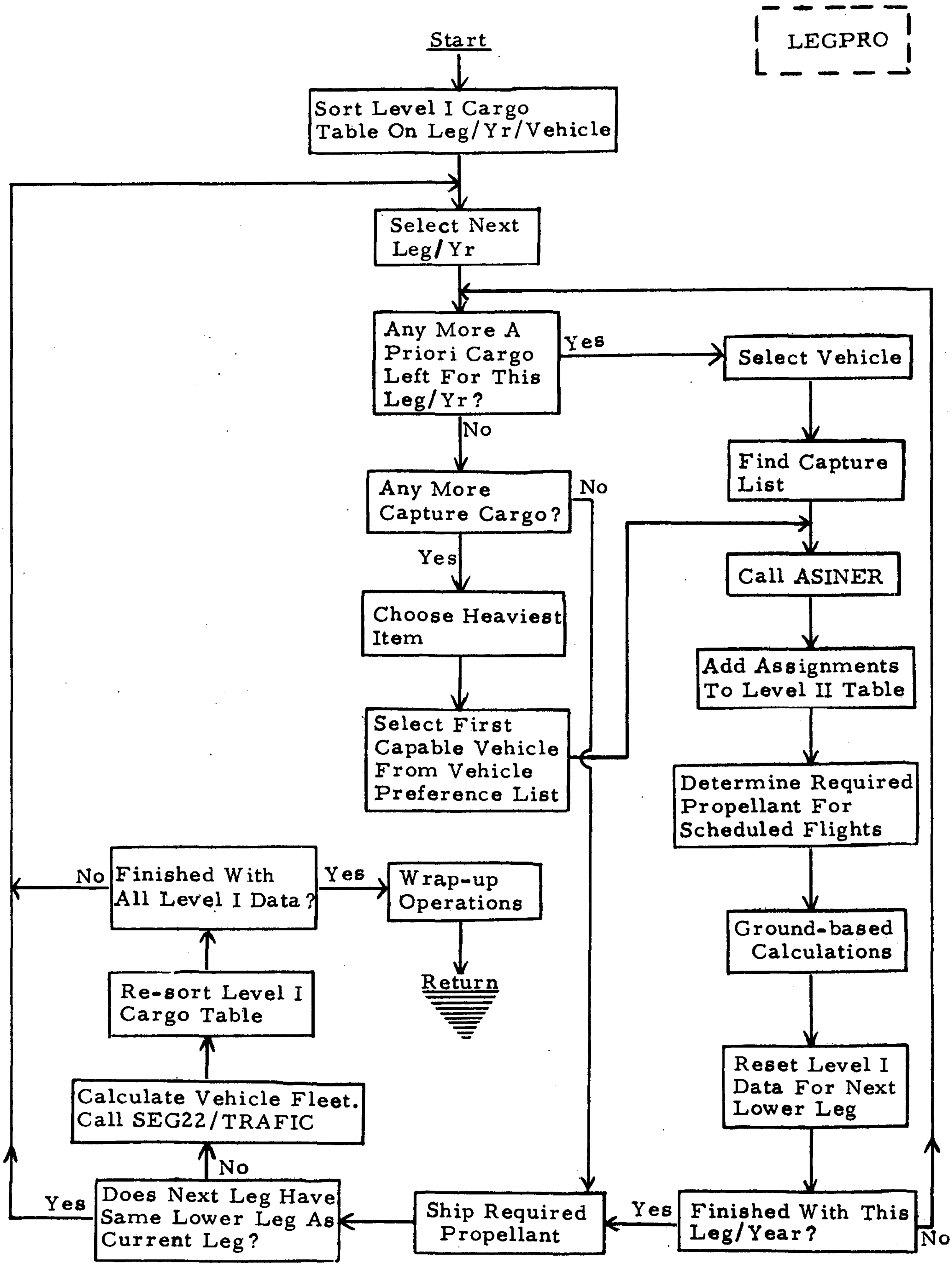
Vehicle calculations

If the automatic vehicle delivery calculation option has been exercised through input, DORCA II will determine how many vehicles of each type must be acquired in addition to those input to fulfill the flights scheduled. This work is performed by subroutine TRAFIC, the call to which is disguised as a call to SEG22. TRAFIC adds these extra acquired vehicles to the vehicle acquisition table and to the Level I cargo table on the next lower leg. Before calling SEG22/TRAFIC, LEGPRO pre-packs the words VEH1 and VEH2 with certain information needed to generate entries in the phase I cargo table. Other information provided for TRAFIC by LEGPRO is the flight table array NFTBL. For each call to ASINER, one entry is added to NFTBL. Each entry is a packed word containing four pieces of information: vehicle index, year, number of flights on current leg, and how many of those flights were expended. Subroutine COLECT collects all entries with identical vehicle index and year, adding up the flights and expended vehicles, into one entry.

Longshoring and Ground-based Operations

If longshoring is permitted at leg nodes, the cargo shipped in containers on one leg can be taken apart and distributed differently. If longshoring is not permitted, then a container or crew capsule plus its contents as prescribed by ASINER on the highest leg must be treated as a discrete, indivisible unit on lower legs. This involves creating a new coupled cargo item in the cargo element table, a discrete item of volume equal to container volume and up/down weight equal to the container weight plus the weights of all parcels inside. A new Level I cargo table entry is created for lower legs; it refers to this new cargo element.

If the ground base option is selected, DORCA II tries to mate the vehicle payload to the vehicle upper stage and treat the combination as a new discrete coupled item. Here also a new entry is created in the cargo element table to represent the upper stage + payload in weight and volume. A new entry in the Level I cargo table refers to this new element. If the combination cannot fit on the vehicles in either or both directions, then LEGPRO will tear it apart and ship the parts separately in either or both directions.


LEGPRO
Start
Sort Level I Cargo Table On Leg/Yr/Vehicle
Select Next Leg/Yr
Any More A Priori Cargo Left For This Leg/Yr?
Yes
Select Vehicle
No
Any More Capture Cargo?
No
Find Capture List
Yes
Call ASINER
Choose Heaviest Item
Add Assignments To Level II Table
Select First Capable Vehicle From Vehicle Preference List
Determine Required Propellant For Scheduled Flights
No
Finished With All Level I Data?
Yes
Wrap-up Operations
Ground-based Calculations
Return
Re-sort Level I Cargo Table
Reset Level I Data For Next Lower Leg
Calculate Vehicle Fleet. Call SEG22/TRAFIC
No
Yes
Does Next Leg Have Same Lower Leg As Current Leg?
Ship Required Propellant
Yes
Finished With This Leg/Year?
No

SECTION 15

SUBROUTINE L2WO

This subroutine is concerned with the storage of the large Level II cargo tables on an external file (tape or disk). The data is stored in logical records of 510 words (except for the last record, which may have less than 510).

L2WO is called from LEGPRO every time a logical record has been formed and is ready to be transferred to "tape" L2WOT. After writing out the data, L2WO sets to zero the number (ICNT) of Level II cargo items still in core and resets the lower and upper limits (NBDDB and NLDDB) of the temporary bucket for the Level II cargo table.

SECTION 16

SUBROUTINE MERGE

PURPOSE

Given a matrix of NC columns and NR rows which is stored out of core on a tape or disk file*, MERGE rearranges the columns so that the elements in row K are in ascending order. An option allows the sort to be either algebraic or alphanumeric. MERGE is used to sort the phase I and II cargo tables.

USAGE

The matrix must be stored at the beginning of FORTRAN logical tape* number NTAPE in binary format, by columns in logical records containing about 510 words. The exact number of columns per logical record (NCPR) is required to be the largest integer not exceeding [510/NR]. If NR is not an exact factor of 510, each full logical record will contain slightly fewer than 510 words. The last logical record should contain only the remaining columns of the matrix and thus will probably contain fewer than NCPR columns. For example, if NR = 3 and NC = 200, then record 1 contains 170 columns (510 words), and record 2 contains 30 columns (90 words). At the conclusion of MERGE, the sorted matrix is replaced on tape NTAPE in exactly the same format.

Under the sort, the columns are rearranged so that elements in row K are in increasing order, either algebraically or alphanumerically, upon option. In the algebraic mode, negative numbers are considered to be less than positive numbers. In the alphanumeric mode, the sign bit is considered to be an overflow from the highest order position; thus, all negative numbers are considered greater than positive ones and are positioned after them at the end of the matrix. This sort option is selected by setting the variable MODE as follows:

```
COMMON /SRTMOD/ MODE
```

where MODE = 0 for alphanumeric sort, MODE $\neq$ 0 for algebraic sort.

---

*In this writeup the word "tape" means either an actual magnetic tape or a disk file.

The calling sequence for MERGE is

CALL MERGE (NTAPE, NC, NR, KELT, MBUFF, FILE, NFILE)

where NTAPE is the logical number of the tape on which matrix is stored.

- NC is the number of columns in the matrix.
- NR is the number of rows in the matrix.
- KELT is the row number on which columns are sorted.
- MBUFF is a temporary 510-word buffer to hold merged data.
- FILE is a matrix of dimension 510 x NFILE to contain several records of unmerged data.
- NFILE is the number of 510-word records which can be stored in FILE. Value is about 10 but depends on spare core which is available to program which in turn depends on the data.

TECHNIQUES

MERGE uses three scratch tapes for intermediate storage of completely sorted partitions of the original matrix. The three tapes are used in circular fashion in that during any stage, one is used for input, one for output, and the third is temporarily idle but contains sorted data. The routine is thus a series of small sort/merges:

Stage 1. Read, sort and merge first NFILE logical records from the input tape and store on scratch tape 1.

Stage 2. Read, sort, and merge next NFILE records from input tape and store on scratch tape 2.

Stage 3, 4, ... Read and sort NFILE-1 records from input tape and merge with contents of one scratch tape onto the free scratch tape. Repeat this stage until the input tape is nearly exhausted (NFILE-2 or fewer records remain), cycling the input/output scratch tapes.

Last stage. Read and sort the remaining NFILE-2 or fewer records from the input tape and merge with the contents of the two scratch tapes last written. Write results onto the rewound input tape. Job is now done.

The algorithm may be illustrated in the following table, where NFILE=10 and an asterisk (*) denotes the data written on that stage:

Stage	Number records read from input tape	Total records on scratch tape 1	2	3
1	10	10*	0	0
2	10	10	10*	0
3	9	0	10	19*
4	9	19*	0	19
5	9	19	28*	0
6	9	0	28	28*
7	9	37*	0	28
8	9	37	37*	0
9	9	0	37*	46*
etc.				

Special provisions are made in the case where the number of logical records is small enough that stage 1 or 2 cannot be completed.

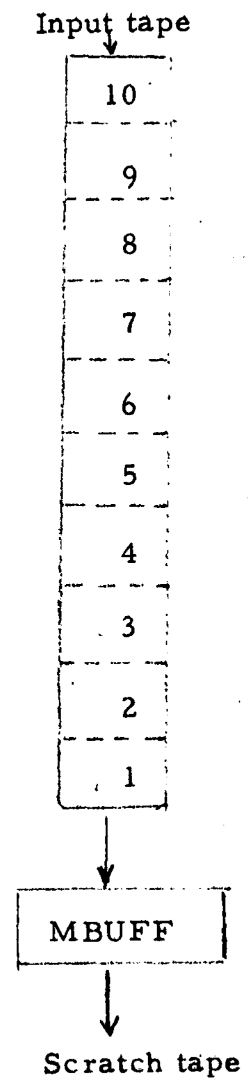
Each of the four tapes used (3 scratch tapes plus the input tape) is assigned an index number for internal use and a corresponding logical number for read/write/rewind operations:

tape	index number	logical number
scratch 1	1	1
scratch 2	2	2
scratch 3	3	3
input	4	NTAPE

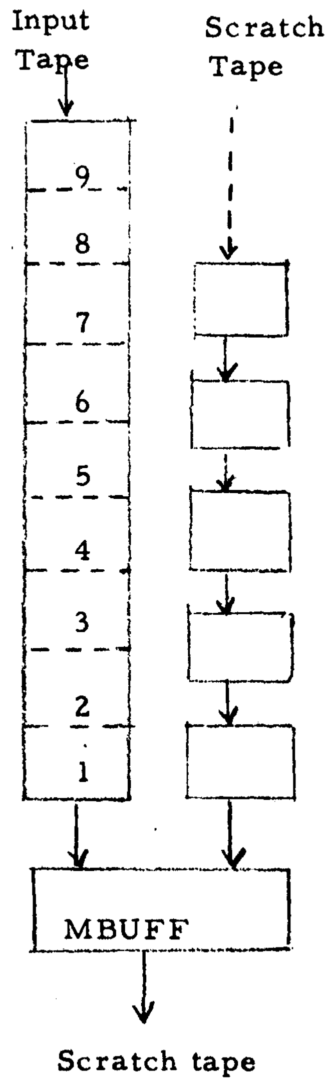
This arrangement allows the logical numbers to be changed with a minimum amount of effort if necessary to avoid conflicts with system or other program use of those logical numbers. A 2 x 4 array TDAT maintains two words of information for each tape - the logical number and the amount of data (number of matrix columns) still residing on the tape. All tape reading and writing is performed in binary mode.

The actual sorting and merging for any stage are done in two arrays - MBUFF and FILE. The merge buffer MBUFF is dimensioned 510 to hold one logical record. As the matrix columns are sorted and merged, they are added to MBUFF; when the buffer is full, it is written to the current output tape.

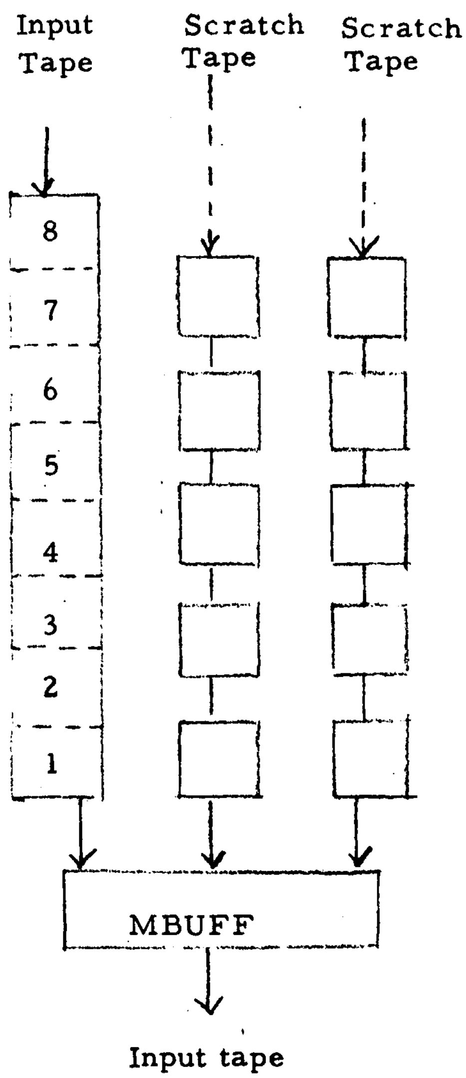
Array FILE is dimensioned NFILE x 510, thus consist of NFILE stacks each the size of a logical record. One or two of these stacks may contain data from intermediate scratch tape(s). During the merge, as soon as all data have been "removed" from one of these stacks to MBUFF, the stack may be refilled by the next logical record on the tape, if any remain. Some or all of the remaining stacks are filled by logical records read from the input tape, which are sorted together into one long stack by subroutine SORT. During the merge, when this stack becomes exhausted it cannot be refilled from the input tape until the next stage. The merge process consists simply in looking at the data at the bottom of each stack, finding the one in which element K is smallest, then pulling the column of data (NR words) out of the stack and adding it to MBUFF. The data is not actually removed from the stack; instead, a set of pointers is maintained in array P indicating the amount of data left in each stack and the position of the "bottom" of the stack.



Stages 1 & 2



Intermediate Stages



Last Stage

Variable	Definition
COLS	Number of columns in matrix FILE which contain data to be merged.
FLOW	Integer used for flow control in assigned GO TO.
I	Index variable.
ICOL	Pointer to column of matrix FILE currently being processed.
IMIN	Number of column containing minimum element and indicating data to be added to merge buffer MBUFF.
INDEX	Index number of scratch tape containing data to be merged.
INTAPE	Logical number of tape/disk file containing original input matrix.
IN1	Index number of first scratch tape containing some data.
IN2	Index number of second scratch tape containing some data.
IP	Pointer to location of data element in minimum test.
ITAPE	Logical tape number of scratch file containing some data.
J	Index variable.
KEL	Subroutine argument indicating column element on which sort is made. Same as subroutine argument KELT.
LNTH	Length of column of matrix FILE (510 unless otherwise changed).
M	Number of matrix columns to read from scratch file.
MINWD	Data element which is minimum so far.
MP	Pointer to next position in merge buffer MBUFF to be filled.
N	Number of matrix columns to be read from input tape.
NCOLS	Number of columns of input matrix still remaining on input tape. Originally set to subroutine argument NC.
NCPR	Number of columns of input matrix per full logical record on input tape.
NFILE	Number of records which FILE can hold.

NREAD — Number of logical records to read from input file into FILE matrix.

NROWS — Number of rows in input matrix. Same as subroutine argument NR.

NWORDS — Number of words to read from input or scratch tape in current logical record.

NWPR — Number of words in a full logical record on input file. (About 510)

OUT — Index of output tape for current merge.

OUTAPE — Logical number of output tape for current merge.

FILE — Matrix containing up to NFILE logical records of data. Each record is stored in a column of FILE (510, 10). As merge progresses, matrix P maintains pointers to FILE and data are removed from FILE and added to the merge buffer MBUFF.

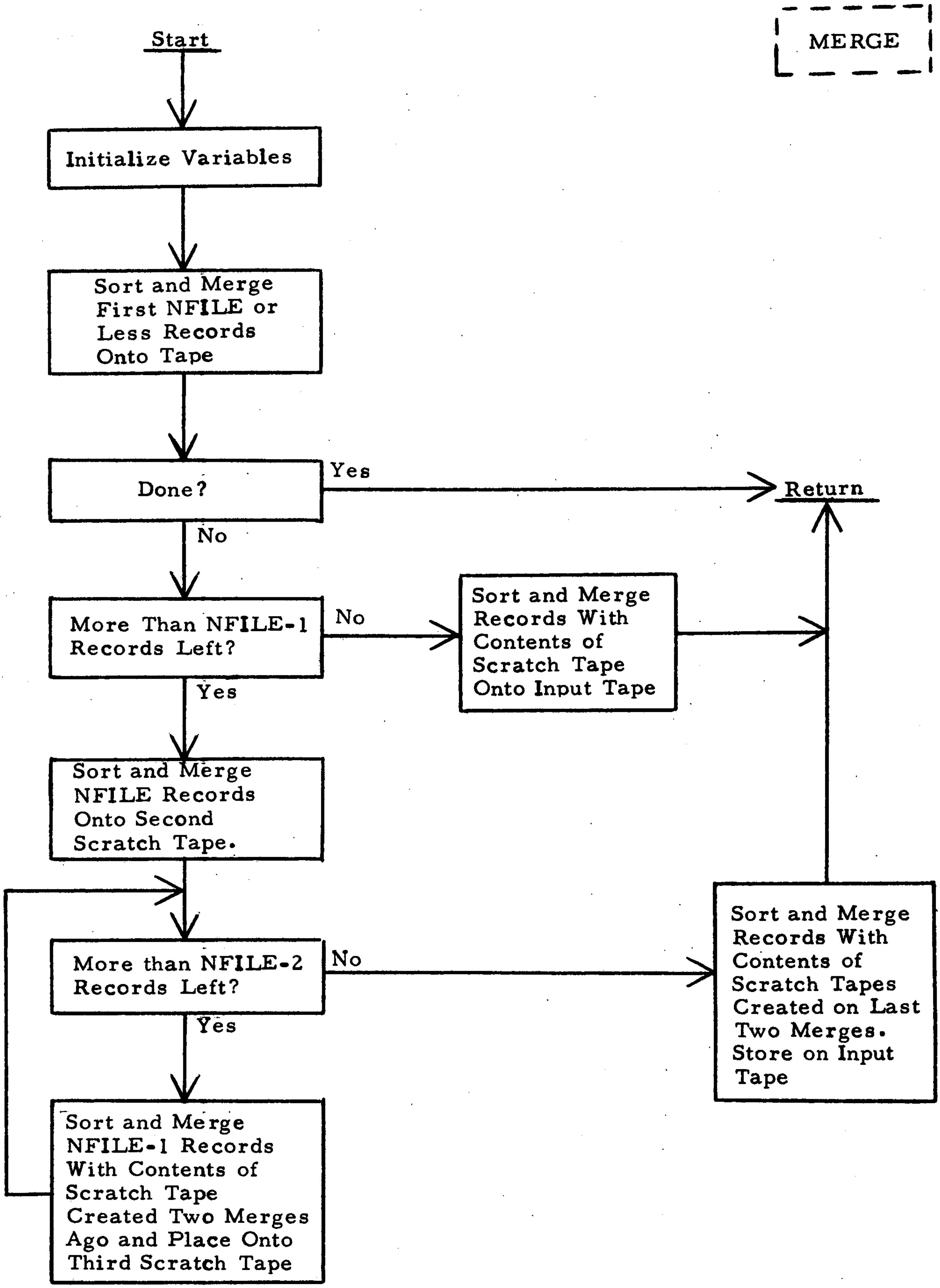
MBUFF — 510-word array containing data merged from columns of FILE. When full, data is written to indicated output file.

P — P(I, J) contains pointers concerning column J of FILE, J = 1, 2, . . . , 10.

- I = 1 : Logical number of tape from which more data is to be read when current data in column J has all been merged into MBUFF. If zero, no more data is to be read into column J when current data are exhausted.
- I = 2 : Number of words that were read into column J.
- I = 3 : Pointer to word of column J which is now being tested for minimum.
- I = 4 : Total number of columns from input matrix still remaining on tape indicated in P(1, J).

TDAT — TDAT(I, J) gives data on file index J, J = 1, 2, 3, 4. J = 1, 2, or 3 are scratch tapes; J = 4 is the input tape.

- I = 1 : Actual logical tape number to be used in read and write statements.
- I = 2 : Number of columns of merged data existing on tape.


MERGE
Start
Initialize Variables
Sort and Merge First NFILE or Less Records Onto Tape
Done?
Yes
Return
No
More Than NFILE-1 Records Left?
No
Sort and Merge Records With Contents of Scratch Tape Onto Input Tape
Yes
Sort and Merge NFILE Records Onto Second Scratch Tape.
More than NFILE-2 Records Left?
No
Sort and Merge Records With Contents of Scratch Tapes Created on Last Two Merges. Store on Input Tape
Yes
Sort and Merge NFILE-1 Records With Contents of Scratch Tape Created Two Merges Ago and Place Onto Third Scratch Tape

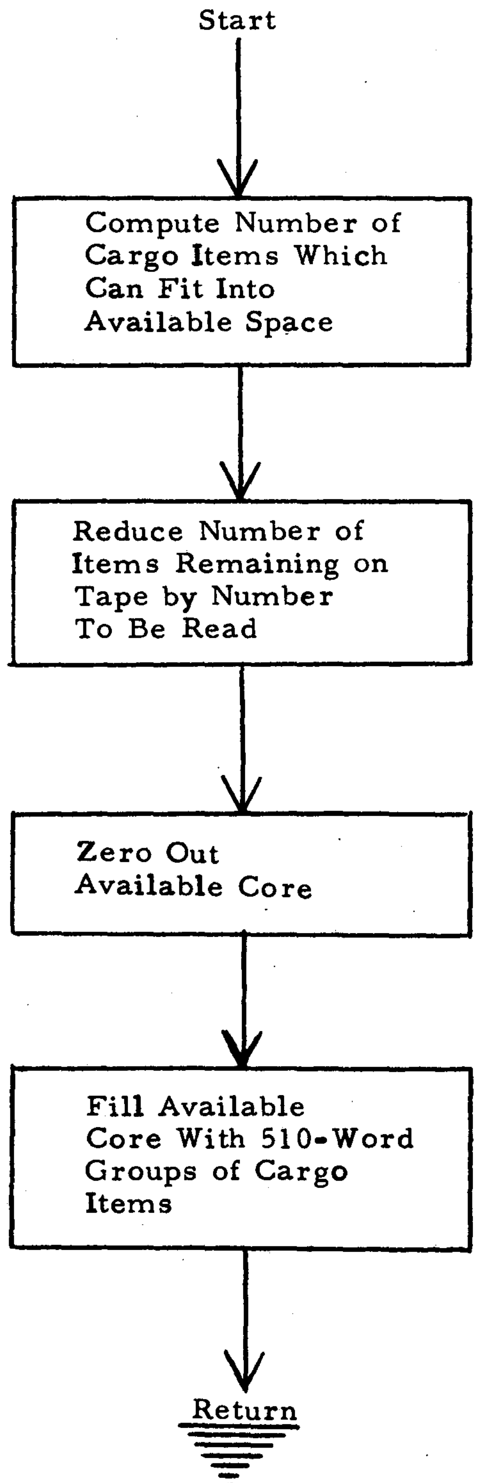
SECTION 17

SUBROUTINE MOREL2

MOREL2 reads more of the Level 2 cargo table from tape L2WOT into core.

MOREL2 is called by VEHRPT, VEHLDF and CSTRPT. The variable NBLK, computed in LEGPRO, contains the number of 510-word blocks which are available in the DDB array to read the cargo table, which has been stored on tape in records of 510 words. The variable MDDB, set to NDDB in the routines which call MOREL2, indicates the number of cargo table items still remaining on the tape (3 words/item, 170 items/record). The routine then reads entire 510-word records, except that the last record on the tape may be shorter than 510.

MOREL2



Start
Compute Number of Cargo Items Which Can Fit Into Available Space
Reduce Number of Items Remaining on Tape by Number To Be Read
Zero Out Available Core
Fill Available Core With 510-Word Groups of Cargo Items
Return

SECTION 18
SUBROUTINE PACK

The purpose of PACK is to insert bits into a given portion of a 36-bit word. PACK isolates the rightmost NBITS bits of variable V and inserts them into packed word PW starting at bit number B. Bit positions are numbered 0 through 35 from left to right.

To pack or insert bits, CALL PACK(V, NBITS, PW, B) where NBITS is the number of bits to be moved ($0 \leq \text{NBITS} \leq 36$).

V is the variable whose rightmost NBITS bits are to be inserted.
PW is the destination word into which the bits will be inserted.
B is the number of the first (leftmost) bit of PW to be replaced. Bits are numbered 0 through 35 from left to right. $B + \text{NBITS} \leq 36$.

Bits of PW other than those from B to B + NBITS - 1 are unchanged. No diagnostics are provided in case of improper values of NBITS and B.

SECTION 19

SUBROUTINE PERLNK

PERLNK computes the performance capabilities for a specific vehicle on a specific leg.

The vehicle, specified by vehicle index NVEH (in common), is the one currently being input to subroutine RDVEH, which calls PERLNK. The velocity increment Δ V required to travel the leg is transmitted in variable VIN in the calling sequence. The variable M describes the expendability of the vehicle:

$$M = \begin{cases} 1 & \text{- totally expandable vehicle} \\ 2 & \text{- expendable upper stage only} \\ 3 & \text{- totally reusable vehicle} \end{cases}$$

PERLNK computes the following variables:

WPLUP - maximum payload up when payload down is zero.
WPLDN - maximum payload down when payload up is zero.
WPEXP - maximum payload up when vehicle is expended.

In addition to the leg velocity increment Δ V, the computations require the specific impulse data for the vehicle. This data is extracted from the vehicle table in DDB and placed in a matrix ARRAY which contains 8 words for each wet (propulsive) stage of the vehicle:

1 WSD (dry structure weight)
2 WNUP (Non-usable propellant weight)
3 WPMAX (Maximum propellant weight)
4 WINT (Interstage weight)
5 WPBO (Boil-off weight)
6 WNIE (Non-Impulsive propellant weight)
7 WACP (Attitude control propellant weight)
8 ISP Number (Specific Impulse)

The algorithm consists of an iteration in which maximum payload is reestimated assuming a linear payload/propellant relationship and then required propellant is computed more exactly by PROPCAL. The initial guess for WPLUP is computed as follows:

$$R = e^{(N \cdot \Delta V)/(G \cdot \Sigma ISP)}$$

$$W_{BO} = WSD_1 + WNUP_1 = \text{ weight of first stage at burnout}$$

$$WPLUP = \frac{WPMAX - WBO\,(R^2-1)}{R-1}$$

where

- N is the total number of propulsive stages
- ΔV is the velocity increment for the leg
- G is the gravitational constant = 32.174 ft/sec^2
- ΣISP is the total specific impulse = sum of specific impulses for each wet stage
- WPMAX is the total propellant available to the vehicle, summed over all stages.

Subroutine PROPCL is now called to compute the propellant required (WPREQ) for a flight on which WPLUP is the up payload and the down payload is zero. Using these values of WPLUP and WPREQ plus the starting values of zero for both up payload and propellant, PERLNK computes a new guess for WPLUP assuming a linear relationship, and the PROPCL is again called to compute a new WPREQ. The iteration continues until the latest value of WPREQ computed by PROPCL is within one pound of total available propellant WPMAX.

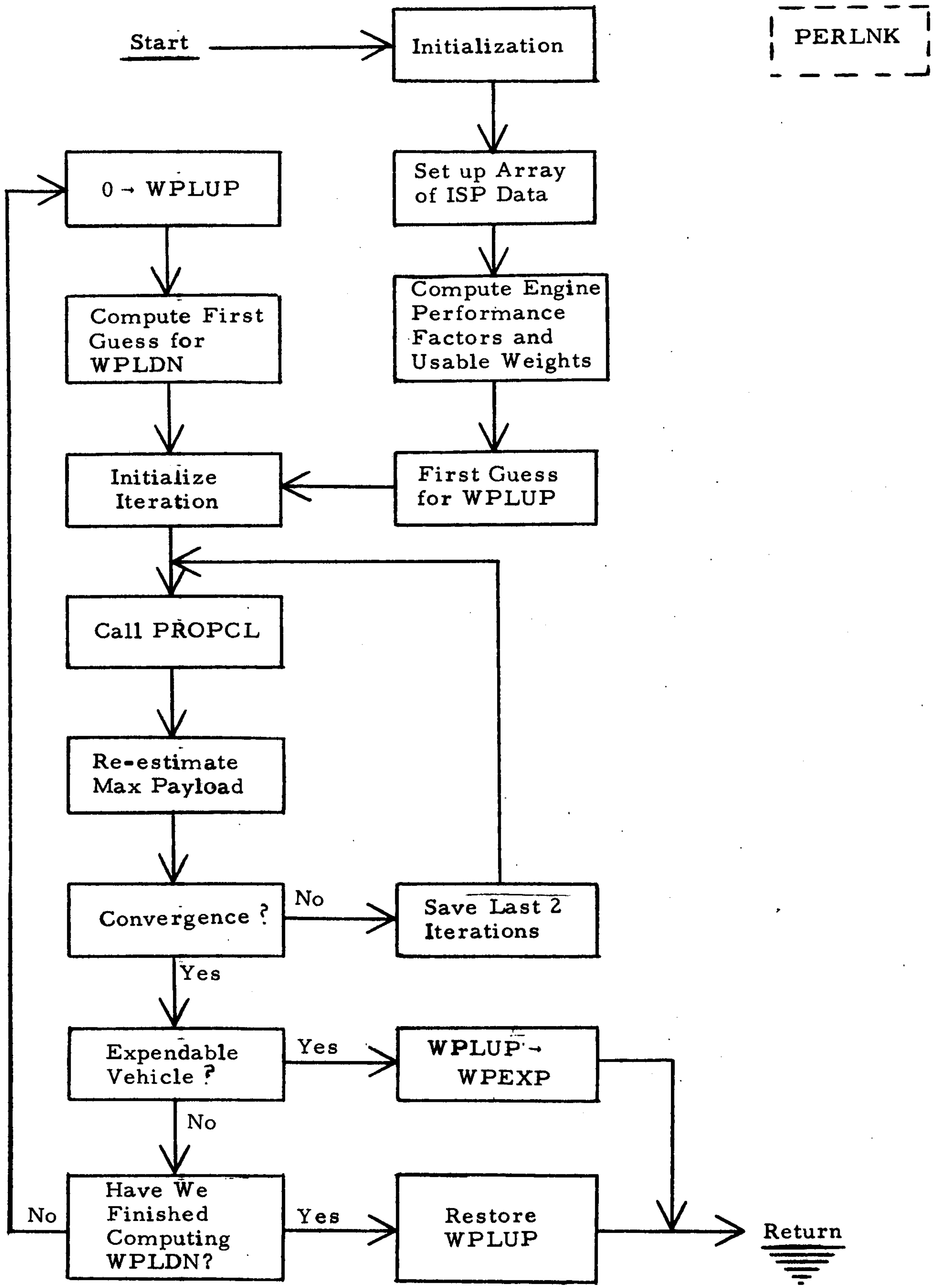
When convergence is attained, the value of WPLUP (which is in PL) is temporarily saved in PLMX to be restored later. Then WPLUP is set to zero and an initial guess for WPLDN is computed:

$$WPLDN = \frac{WPMAX - WBO\,(R-1)}{R^2 - R}$$

where R and WBO are as above. Then the iteration is repeated until convergence.

If $M \neq 3$ (indicating an expendable vehicle), then WPEXP is taken as the converged value of WPLUP, and the computation for WPLDN is omitted.

PERLNK is called from RDVEH once with $M = 3$ if WPLUP/WPLDN were <u>not</u> input, and once with $M = 2$ if WPEXP was not input.


PERLNK
Start
Initialization
Set up Array of ISP Data
Compute Engine Performance Factors and Usable Weights
First Guess for WPLUP
0 → WPLUP
Compute First Guess for WPLDN
Initialize Iteration
Call PROPCL
Re-estimate Max Payload
Convergence ?
No
Save Last 2 Iterations
Yes
Expendable Vehicle ?
Yes
WPLUP → WPEXP
No
Have We Finished Computing WPLDN?
No
Yes
Restore WPLUP
Return

SECTION 20

SUBROUTINE PROLNK

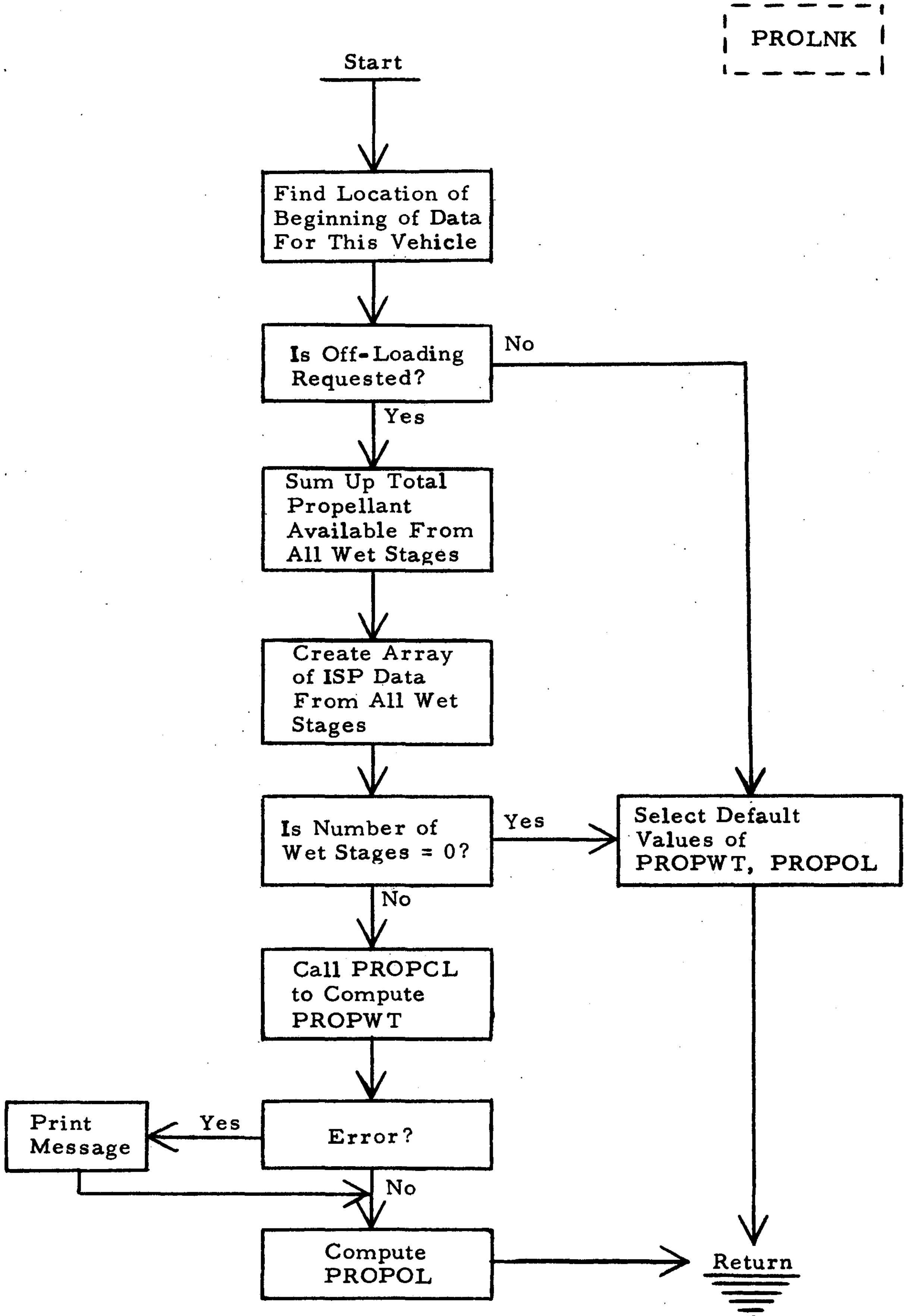
PROLNK is a link between LEGPRO and PROPCL in determining propellant requirements and offloading if required for a specific vehicle flight.

LEGPRO calls PROLNK with a given vehicle, up and down payload weights, and velocity increment for the leg flight. PROLNK is to compute the total propellant PROPWT needed for the flight and the amount off-loaded PROPOL into the first stage. If propellant off-loading is not allowed (i.e., vehicle must carry full amount of propellant whatever the payload weight), or if the vehicle has no wet stages, then PROPWT $\equiv$ PROPOL $\equiv$ total vehicle propellant capacity.

If off-loading is required, PROPCL sets up a matrix array of 8 words of ISP data for each stage in the vehicle, obtained from the vehicle table. Subroutine PROPCL is called to compute PROPWT. The total propellant, PROP, available to the vehicle, is computed as the sum of the propellants for each stage. Then PROPOL is simply the first (lowest) stage propellant reduced by the amount by which total propellant available exceeds propellant required:

$$\text{PROPOL} = \text{PROP}_1 - (\text{PROP} - \text{PROPWT})$$

That is, only the first stage is off-loaded.


PROLNK
Start
Find Location of Beginning of Data For This Vehicle
Is Off-Loading Requested?
No
Yes
Sum Up Total Propellant Available From All Wet Stages
Create Array of ISP Data From All Wet Stages
Is Number of Wet Stages = 0?
Yes
Select Default Values of PROPWT, PROPOL
No
Call PROPCL to Compute PROPWT
Print Message
Yes
Error?
No
Compute PROPOL
Return

SECTION 21

SUBROUTINE PROPCL

PROPCL calculates the propellant required, WPREQ, for a specified vehicle flight on a given leg. This calculation is used for computing vehicle performance capabilities when called from RDVEH/PERLNK, and for determining propellant offloading when called from LEGPRO/PROLNK.

The leg data passed to PROPCL through the calling sequence consists of the velocity increment VIN (or Δ V) necessary to leave the orbit at the bottom of the leg and achieve the orbit at the top. The vehicle data consists of the following:

NTOT = number of propulsive stages comprising vehicle;
WPLUP = weight of payload to be carried up;
WPLDN = weight of payload down

$$M = \begin{cases} 1 & \text{- totally expendable vehicle} \\ 2 & \text{- expendable upper stage only} \\ 3 & \text{- totally reusable vehicle} \end{cases}$$

S = a sort of "safety" factor used in engine thrust calculations which in DORCA is always, taken as 1.0 .

ARRAY = a matrix of 8 words for each of the NTOT propulsive stages:

WORD	CONTENTS.
1	WSD (dry structure weight)
2	WNUP (Non-usable propellant weight)
3	WPMAX (Maximum propellant weight)
4	WINT (Interstage weight)
5	WPBO (Boil-off weight)
6	WNIE (Non-Impulsive propellant weight)
7	WACP (Attitude control propellant weight)
8	ISP Number (Specific Impulse)

PROPCL also computes NREQ, the number of stages of the vehicle actually required to achieve the necessary thrust. Ideally, NREQ should equal NTOT. If NREQ > NTOT, the vehicle is inadequate for the flight. If NREQ < NTOT, not all stages are necessary.

ALGORITHM

PROPCL first calculates four variables for each propulsive stage:

WB_i = weight of stage i at burnout
 = dry structure + nonusuable propellant + interstage weight

$WPMX_i$ = maximum usable propellant
 = maximum propellant - nonusable propellant

ISP_i = specific impulse of stage i

$ISPEF_i$ = effective specific impulse

$$= ISP_i \cdot \left(\frac{WPMX_i - \text{propellant losses}}{WPMX_i} \right)$$

where propellant losses are boil-off weight, non-impulsive propellant, and attitude-control propellant.

The algorithm starts with the last (highest) stage (N=NTOT). VRQ is the total velocity required from this and all remaining (lower) stages; initially, PROPCL sets VRQ = VIN and reduces it by the velocity V available from each stage. WE is the weight of the vehicle at the end of burnout of each stage, a running total which accumulates with each stage.

There are two sections to the algorithm - the flight up the leg, and the flight down, starting at the bottom of the down flight, working backwards to the top of the leg and then backwards down the up flight. If the vehicle is totally reusable (M = 3), both sections are processed; if the vehicle is partly or wholly expendable, the first section (down flight) is omitted.

For each section, the basic steps of the computation are:

(1) Compute velocity V available to stage N:

$$V = G \cdot ISP_{eff_N} \cdot \frac{\ell n \,(1 + WPMX_N \,/\, WE)}{S}$$

where G is the gravitational constant 32.174 ft/sec^2 and $S \equiv 1$ in DORCA.

(2) If $V \geq$ required velocity VRQ, this section is finished; exit.

(3) If $V < VRQ$, reduce VRQ by V.

(4) If $N = 1$, vehicle is inadequate for this flight; Error exit, if $N > 1$, go to next lower stage. $N-1 \rightarrow N$

(5) Compute propellant required at this stage:

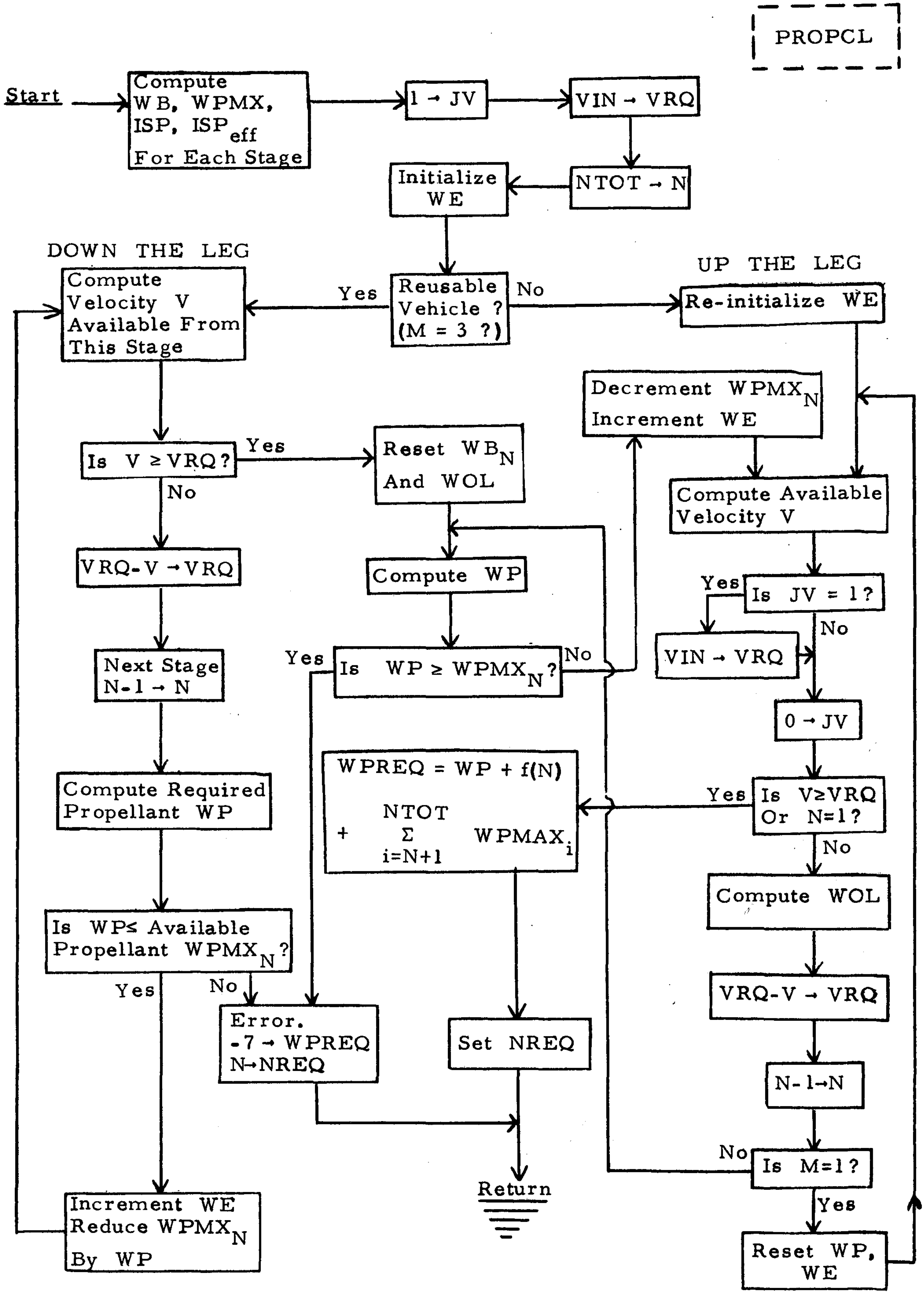
$$WP = WB_N \left[e^{f(N)} - 1.0 \right]$$

where $f(N) = \dfrac{S\ (VIN - VRQ)}{G \cdot ISP_{eff_N}}$

(6) If $WP >$ available usable propellant $WPMX_N$, vehicle is inadequate. Set $WPREQ = -7$ as a signal and return.

(7) If $WP \leq WPMX_N$, continue algorithm.
Reduce $WPMX_N$ by WP, accumulate burnout weight and propellant into WE, and go to step (1).

When the required velocity is attained, the propellant requirements for each stage are summed up into WPREQ.


PROPCL
Start
Compute WB, WPMX, ISP, ISP eff For Each Stage
1 → JV
VIN → VRQ
NTOT → N
Initialize WE
DOWN THE LEG
UP THE LEG
Reusable Vehicle ? (M = 3 ?)
Yes
No
Compute Velocity V Available From This Stage
Re-initialize WE
Decrement WPMX N
Increment WE
Is V ≥ VRQ ?
Yes
No
Reset WB N And WOL
Compute Available Velocity V
VRQ-V → VRQ
Compute WP
Yes
Is JV = 1 ?
No
VIN → VRQ
Next Stage N-1 → N
Yes
Is WP ≥ WPMX N ?
No
0 → JV
WPREQ = WP + f(N) + Σ (i=N+1 to NTOT) WPMAX i
Yes
Is V≥VRQ Or N=1 ?
No
Compute Required Propellant WP
Compute WOL
Is WP≤ Available Propellant WPMX N ?
Yes
No
VRQ-V → VRQ
Error. -7 → WPREQ N→NREQ
Set NREQ
N-1→N
No
Is M=1 ?
Yes
Return
Increment WE Reduce WPMX N By WP
Reset WP, WE

C-2

SECTION 22
SUBROUTINE RDCONT

RDCONT is called by INPRO to read, process and store container data.

Before processing the first card, the entire container table TBCONT is zeroed out. Cards are read as 8 fields of 10 characters each and stored as 8 pairs in array CARD in A6, A4 format. The order of variables on each card is

Field 1 - container name
Field 2 - capacity
Field 3 - empty weight
Field 4 - classification (bulk, crew, or propellant)
Field 5 - volume
Field 6 - expend option

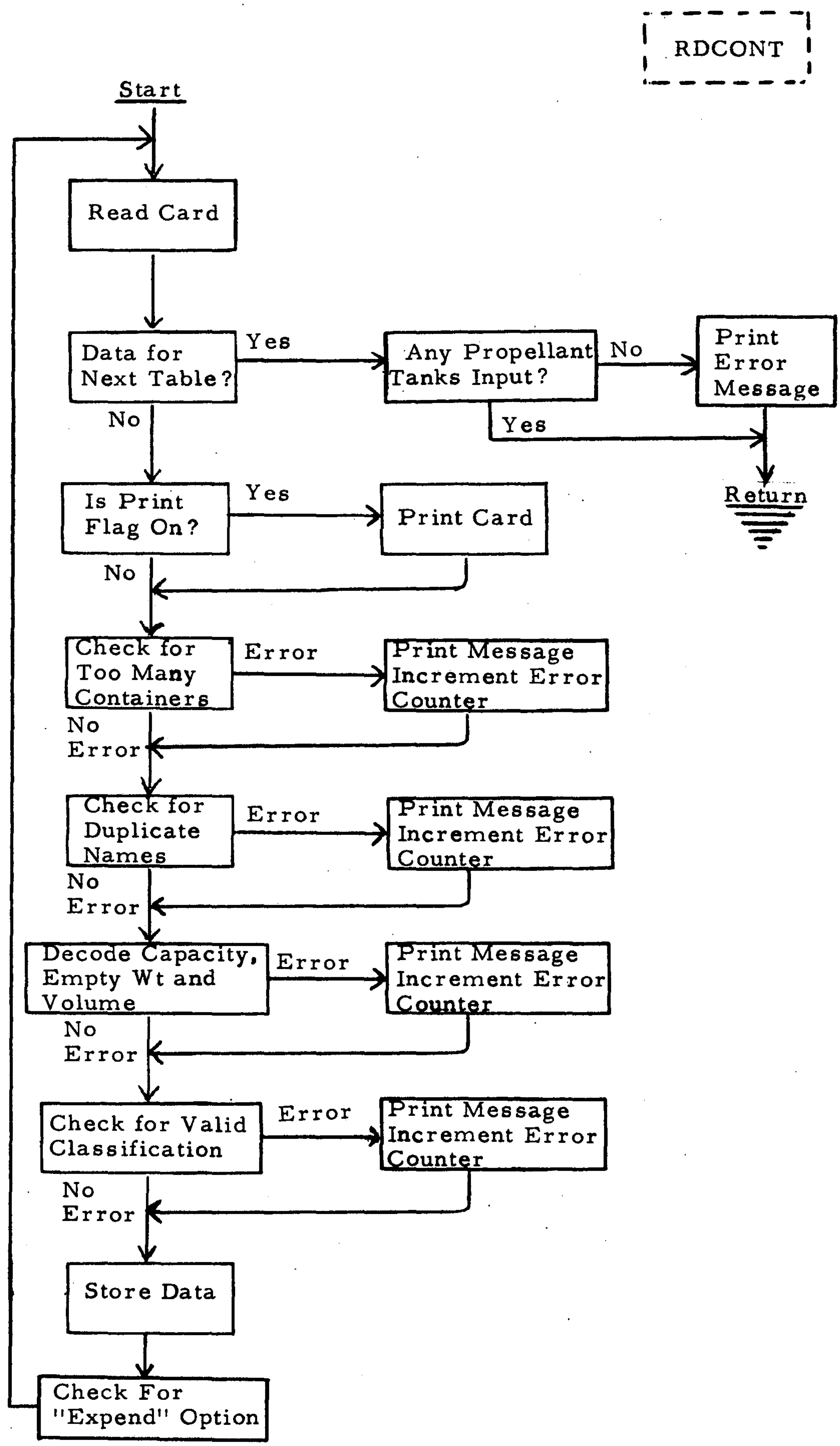
The numerical entries (capacity, empty weight and volume) are converted from coded to floating point format by subroutine VALUE. NCONT is the number of containers read so far. Data for container number J ($1 \leq J \leq NCONT$) are stored as follows:

TBCONT(1, J) - name (first 6 letters)
TBCONT(2, J) - name (last 4 letters)
TBCONT(3, J) - capacity weight
TBCONT(4, J) - empty weight
TBCONT(5, J) - classification (1.0-crew, 2.0-bulk, 3.0-not used, 4.0-propellant)
TBCONT(6, J) - volume (default value is 1.0 if not input)
TBCONT(7, J) - used for a flag in CNTRPT
TBCONT(8, J) - return/expend option

For proper bookkeeping, at least one propellant tank should be input. The variable JPROP is set to the index number J of the (last) propellant tank input and variable PROPUP is set to its capacity. This data is used to modify container data in subroutine RDCRG. If no propellant tank is input, an error message is printed.

RDCONT continues reading input cards one at a time, storing data and printing error messages when necessary, until the word TABLE is encountered in field 1. This word signifies the end of container data, and control is returned to subroutine INPRO. The following input errors will generate printed messages:

1. Duplicate container names.
2. Invalid numerical entry for capacity, empty weight or volume.
3. Improper word entry for classification.
4. Too many containers input. Maximum number of containers if given by the variable NCTMAX, which presently has the value 20.
5. No propellant tank input.


RDCONT
Start
Read Card
Data for Next Table?
Yes
No
Any Propellant Tanks Input?
No
Yes
Print Error Message
Return
Is Print Flag On?
Yes
No
Print Card
Check for Too Many Containers
Error
No Error
Print Message Increment Error Counter
Check for Duplicate Names
Error
No Error
Print Message Increment Error Counter
Decode Capacity, Empty Wt and Volume
Error
No Error
Print Message Increment Error Counter
Check for Valid Classification
Error
No Error
Print Message Increment Error Counter
Store Data
Check For "Expend" Option

SECTION 23

SUBROUTINE RDCRG

This subroutine reads and stores cargo elements.

Data for a single cargo element occupies one card. As described in the user's manual, the input card format is slightly different than for the other input routines:

Card Columns	Field	Contents
1-10	1	Cargo element name (10 characters)
11-28	2	Description (18 characters)
29-30	-	(Not used)
31-40	3	Container
41-50	4	Category
51-60	5	Up weight
61-70	6	Down weight
71-80	7	Volume

Field 2 (description), allotted 18 characters, is the only item not using a standard 10-character field. It is stored in 3 words in (A6, A6, A6) format.

The routine calls READ to read one card at a time. If the first field contains the word TABLE or PROGRAM, control is returned to subroutine INPRO to process the next data table. Otherwise, the contents of field 1 are taken as the name of the next cargo element. Cargo elements are stored in array DDB in groups of 9 words as follows:

Word	Contents
1-2	Name (A6, A4 format)
3-5	Description (A6, A6, A6 format)
6	Packed word:
	Bits 0-11: Pointer to vehicle or facilities table (see below)
	Bits 12-23: Container class (1-crew, 2-bulk, 3-discrete, 4-propellant, 5-coupled item)
	Bits 24-35: Category (1-material, 2-personnel, 3-facilities and satellites, 4-vehicle)

Word	Contents
7	Up weight (floating point)
8	Down weight (floating point)
9	Volume (floating point). Used only for discrete items, zero for others.

In order to determine the 3 variables (pointer, class, and category) in the packed word, RDCRG first checks the name entered in field 3 (container field) against all names in the container table TBCONT. If a match is found, the pointer is set to the index of the container in TBCONT and the class is determined by the container data (1-crew, 2-bulk, 4-propellant). If no match is made, the field should contain the word DISCRETE, in which case the class is 3 and the category is determined by the entry in field 4. The cargo element name is compared with the vehicle names in the vehicle table TBVEH and the facility names in the facility table (stores in DDB) until a match is made, and the pointer is set to the number of that vehicle or facility. If the names on this card cannot be matched with an entry in the container, vehicle or facility tables, an error message is printed.

RDCRG is called from INPRO. If the variable NAFAC = 0 upon entry, RDCRG is supposed to process the entire cargo element table - that is, to continue reading and store new cargo elements until a card is read with the key word "PROGRAM" or "TABLE" in field 1. If NAFAC = 2 upon entry, RDCRG will process only the single card whose image appears in array CARD, then returns. This situation represents the appearance of an ELEMENT card within the mission data being processed by RDMISS.

When an input card is read with the word TABLE in field 1, all data in the cargo table have been read. However, containers in the container table already input will also form part of the cargo. Therefore, before returning control to INPRO, RDCRG adds all containers to the end of the cargo table, considering them discrete items and storing them in the same 9-word group as other cargo elements. The parameters are set as follows:

Name $\equiv$ description = container name

Pointer = 0

Class = 3 (discrete)

Category = 1 (material)

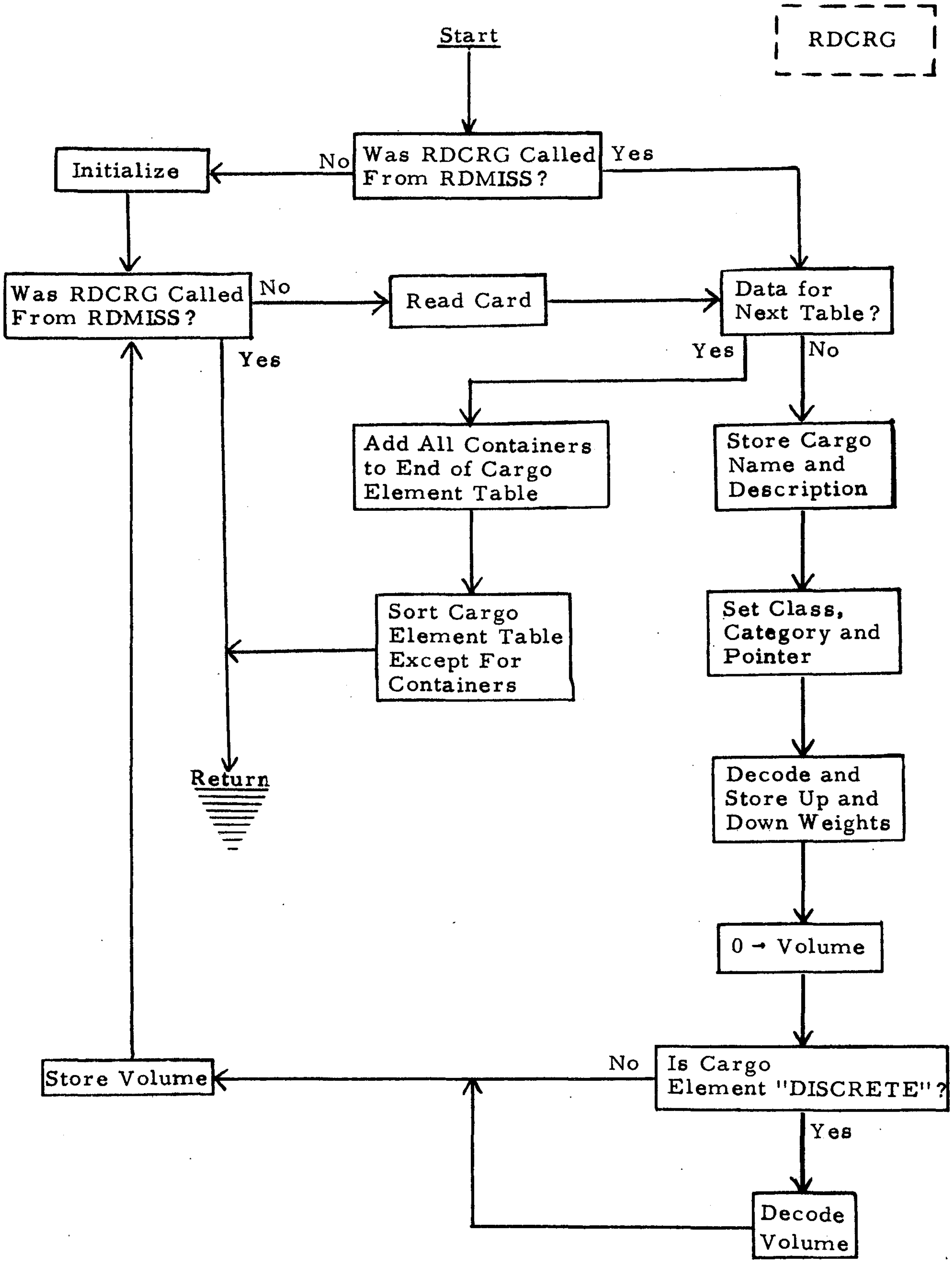
Up weight $\equiv$ down weight = container empty weight (except see below)

Volume = container volume

This treatment is useful for shipping empty containers. The one exception to this treatment is propellant tanks; these are added to the cargo element table with up weight = container capacity + container empty weight, so that propellant tanks can be shipped upwards full and downwards empty as discrete items.

An artificial device has been installed which forces the program to print a cost report on any or all of the containers specified by the user. The device involves treating containers as though they were also facilities. The user must input to the facility table dummy facilities whose names are the same as those of the containers to be costed. For these particular containers, the program will set CAT = 3 and PNTER to the index in the facility (not container) table.

Before returning INPRO, RDCRG sorts the cargo element table (not including containers) into alphabetical order.


RDCRG
Start
Was RDCRG Called From RDMISS?
No
Yes
Initialize
Was RDCRG Called From RDMISS?
No
Yes
Read Card
Data for Next Table?
Yes
No
Add All Containers to End of Cargo Element Table
Store Cargo Name and Description
Sort Cargo Element Table Except For Containers
Set Class, Category and Pointer
Return
Decode and Store Up and Down Weights
0 → Volume
Store Volume
No
Is Cargo Element "DISCRETE"?
Yes
Decode Volume

SECTION 24
SUBROUTINE RDFAC

RDFAC reads and stores facility data.

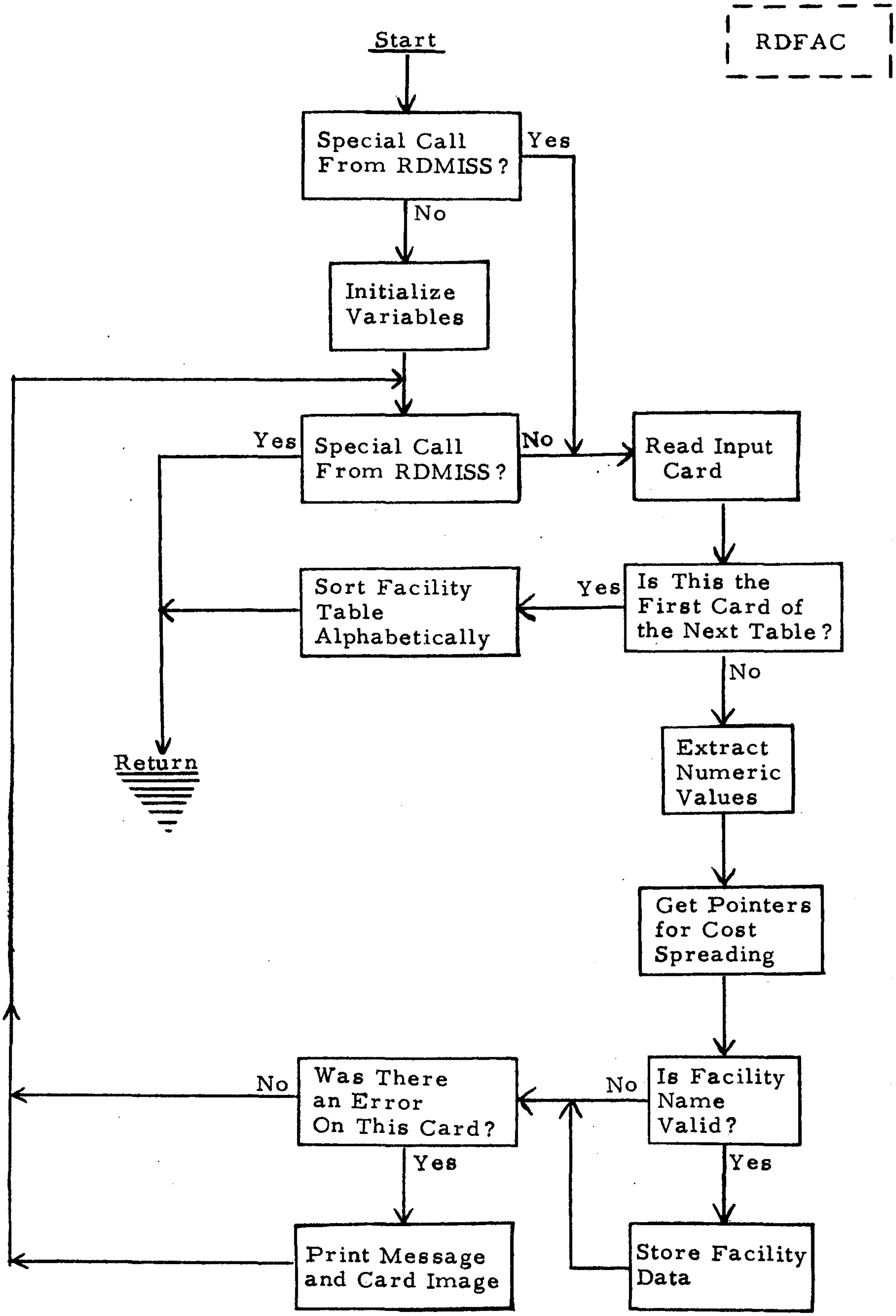
The coding is straightforward. Data cards are read as 8 fields of 10 characters each; each 10-character field is stored in array CARD as a pair of alphanumeric coded words in (A6, A4) format. For this subroutine, only 6 fields are used on each card for the 6 entries - name, years of life, development cost, production cost, and names of spread functions to be applied to production and development costs. Numeric entries (years of life and costs) are converted to floating point by subroutine VALUE. Subroutine FINDSP is called to find pointers JSPD and KSPD in the spread table corresponding to the input spread function names; JSPD and KSPD are in floating point format, despite having integer names.

RDFAC is called from INPRO. If the variable NAFAC = 0 upon entry, RDFAC is supposed to process the entire facility table -- that is, read cards and store facilities until a card appears with the key word "TABLE" or "PROGRAM" in field 1, signalling the end of the facility table.

If NAFAC = 1 upon entry, RDFAC is to process only the single card whose image appears in array CARD, then return to INPRO. This situation represents the appearance of a FACILITY card within the mission data being processed by RDMISS.

Data are stored in DDB(I), (I = NBFAC, ..., NLFAC), in groups of 8. All items except the facility name are in floating point, including spread function pointers. The order within each group of 8 is:

1-2	Facility name (A6, A4) format
3	Life span in years
4	Development cost
5	Pointer to spread table for development cost
6	(Not used)
7	Recurring production cost
8	Pointer to spread table for production cost


RDFAC
Start
Special Call From RDMISS?
Yes
No
Initialize Variables
Special Call From RDMISS?
Yes
No
Read Input Card
Sort Facility Table Alphabetically
Yes
Is This the First Card of the Next Table?
No
Return
Extract Numeric Values
Get Pointers for Cost Spreading
No
Was There an Error On This Card?
No
Is Facility Name Valid?
Yes
Yes
Print Message and Card Image
Store Facility Data

SECTION 25

SUBROUTINE RDLEG

RDLEG reads leg data and stores it in matrix TBLEG (which is later altered in subroutine RDVEH).

RDLEG reads (by calling READER) and processes one input card at a time until a card is encountered whose first field contains either of the words TABLE or PROGRAM. At the time, the input leg data is finished and control is returned to the calling routine, INPRO. If field 1 contains anything else, that entry is taken as the name of a leg. The input card format is:

Field	Contents
1	Leg name
2	Lower (previous) leg name
3	Deployment limit for any vehicle on this leg
4	Default vehicle name
5	Δ V
6	Alternate vehicle
7	Longshoring option
8	(not used)

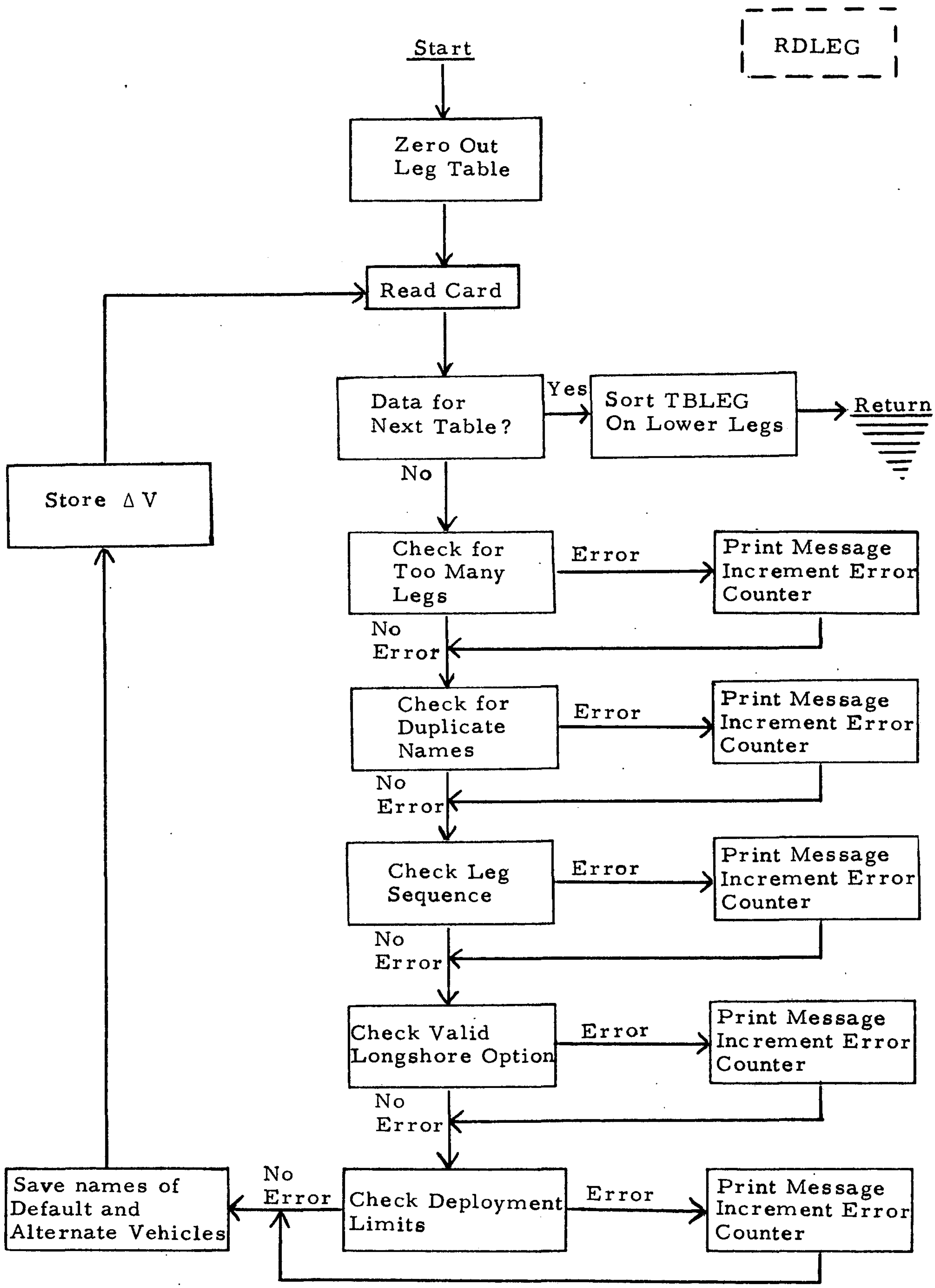
Leg data is stored in matrix TBLEG(I, J) (J = 1, 2, ..., 12) as follows:

I	Format	Contents of TBLEG (I, J)
1-2	A6, A4	Leg name.
3-4	A6, A4	Name of lower (previous) leg
5	real	Deployment limit
6-7	A6, A4	Default vehicle name
8	-	Temporary use
9	real	Δ V (floating point)
10	real	Longshoring flag (0-no, 1-yes)
11-12	A6, A4	Alternate vehicle name

The contents of TBLEG are modified further in subroutine RDVEH.

When all legs have been read in, TBLEG(B, J) is set to the index number of the next lower previous leg, is determined from TBLEG (3-4, J) for each J. If the next lower leg is "NONE", TBLEG(8, J) is set to the key word LASTLG, whose current key value is 63. Then the entire matrix is sorted on word 8. This entire procedure, including the sort, is performed twice so as to arrange the leg names in a manner which is most convenient for the programmer and user reading the printout generated by DORCA II.

The maximum number of legs which can be accomodated is NLGMAX $\equiv$ 62 (63 including the special leg named "NONE").


RDLEG
Start
Zero Out Leg Table
Read Card
Data for Next Table?
Yes
Sort TBLEG On Lower Legs
Return
No
Store ΔV
Check for Too Many Legs
Error
Print Message Increment Error Counter
No Error
Check for Duplicate Names
Error
Print Message Increment Error Counter
No Error
Check Leg Sequence
Error
Print Message Increment Error Counter
No Error
Check Valid Longshore Option
Error
Print Message Increment Error Counter
No Error
Save names of Default and Alternate Vehicles
No Error
Check Deployment Limits
Error
Print Message Increment Error Counter

SECTION 26
SUBROUTINE RDMISS

RDMISS reads the program/mission data, stores it as the phase I cargo table, and generates the initial vehicle and facility acquisition tables.

Standard Mission Input

The input format for program and mission data is given in the user's manual. Briefly, each program/mission involves shipping a certain number of units of a given cargo element in a given vehicle, leg, and year(s). Associated with this data is a phase (initializing, sustaining or terminating). Also associated is a list of up to three vehicles to be used on lower legs, if any. The vehicles used on this and lower legs are obtained from the leg table TBLEG if not input as part of the mission data.

Each one of the units of a cargo element shipped generates a trio of words in the phase I cargo table. The format of these three words is given in Appendix B.

The name of each program is stored in matrix PNAME, 18 characters per name in (A6, A6, A6) format, with a maximum of 63 names. The program name OVERHEAD is built into the routine. Similarly, the name of each mission is stored in matrix MNAME, 18 characters per name in (A6, A6, A6) format, with a maximum of 63 names. Three mission names, PROPELLANTS, VEHICLES, and CONTAINERS, are built in.

For each cargo element shipped, the category is examined. If the element is a vehicle or facility, an additional entry is made in the vehicle or facility acquisition table. The format of these tables is given in Appendix B.

Much of the labor of RDMISS consists of trying to match input entries with previously input names in the container, log, vehicle, cargo element,

3

and mission and program name tables. When matches are made, the input entries can thus be converted to index numbers referring to the tables. Subroutine VALUE is called to convert numeric input -- dates, mission phase (which may be either numeric or alphabetic), cargo multiplicity number -- from coded to floating point format. As the information is accumulated, it is packed into a 3-cell array, ICARGO, which is eventually transferred into DDB to become part of the cargo table. The format of ICARGO is the same as that of the phase I cargo table (see Appendix B).

When the routine detects an input error, a message is printed and the error counter JERR is incremented by 1. The routine will continue processing all cards as best it can, but each error will mean that some data or default values are stored which may be incorrect. The final reports will therefore contain some incorrect data. The values actually stored for an erroneous entry are usually the last correct entry of that type. The intent of this approach is to execute the program as far as possible to uncover as many errors as possible in one run.

Certain key words which may occur in field 1 (columns 1-10) of the input card are recognized:

PROGRAM MISSION PHASE IOC LEG VEHICLE STOP START CARGO	standard mission input
COUPLE	coupled items
FACILITY ELEMENT	new entries into facility or cargo element table
SCHEDULE SATELLITE	schedule or satellite shipping data
REPORT	end of mission data

Coupled Items

The COUPLE feature permits the user to specify that two or more previously input cargo elements are to be combined into a box that will be treated as a unit. This box will be a new cargo element. The card format for each coupled unit is:

Card #	Field 1	Field 2
1	COUPLE	Name of new box
2	Element 1	
3	Element 2	
.	.	
.	.	
.	.	
M+1	Element M	
M+2	END	

where "Element 1", etc., denotes the name of a previously input cargo element.

For each such element named, RDMISS locates that name in the cargo element table and determines its index N in that table. A list of the indices N for each of these M elements is saved in the array CE, between the limits of I1 and IN. That is

$$N = CE(K), \quad K = I1, \ldots, IN$$

is a list of the component elements comprising the box. The up and down weights and volume of this box is the sum of those quantities for the individual elements. The entry created in the cargo element table for this box has the standard format:

Word	Contents
1	Name of new box (first 6 characters)
2	Name of new box (last 4 characters)
3	Description = the word "COUPLE"

Word	Contents
4	I1 (see above)
5	IN
6	Octal packed word 000000050001 (means pointer = 0, class = 5, category = material)
7	Total up weight
8	Total down weight
9	Total volume

For costing purposes, the routine computes a weight load factor for each component. This is defined as

$$\text{load factor} = 100000 \times \frac{\text{component weight}}{\text{total coupled item weight}}$$

The component weight is taken as the up weight unless that is zero, in which case down weight is used. The multiplier 100000 is used to make the most important digits whole numbers and eliminate fractions so that the load factor can be converted to integer format. It is stored temporarily in the array WLF, which is indexed the same as array CE, and then packed into the level I cargo table

Facility and Element Input

Additional facilities or cargo elements may be input in the midst of the mission data. A card with the word FACILITY in field 1 defines the following card(s) as facilities rather than mission data. RDMISS sets NAFAC = 1, reads another card, then returns to INPRO, which calls RDFAC to process that single card and enter the data into the facility table. Then RDMISS is called again to read another card; if field 1 of the new card does not contain one of the key words, the card is presumed to be another facility card, and the procedure above is repeated.

A card with the word ELEMENT in field 1 defines the following card(s) as new cargo elements. The process is similar to that for facilities except that NAFAC = 2 and INPRO calls RDCRG to process each card singly.

Whenever RDMISS finds NAFAC = 1 or 2, DDBSFT is called to shift data in the DDB array if necessary to accomodate the new cargo elements or facilities.

SCHEDULE AND SATELLITE INPUT

SCHEDULE and satellite data may be input for non-consecutive IOC/cargo dates. Schedule is used when different cargo are on the same vehicle; satellite is used when the same cargo is on different vehicles.

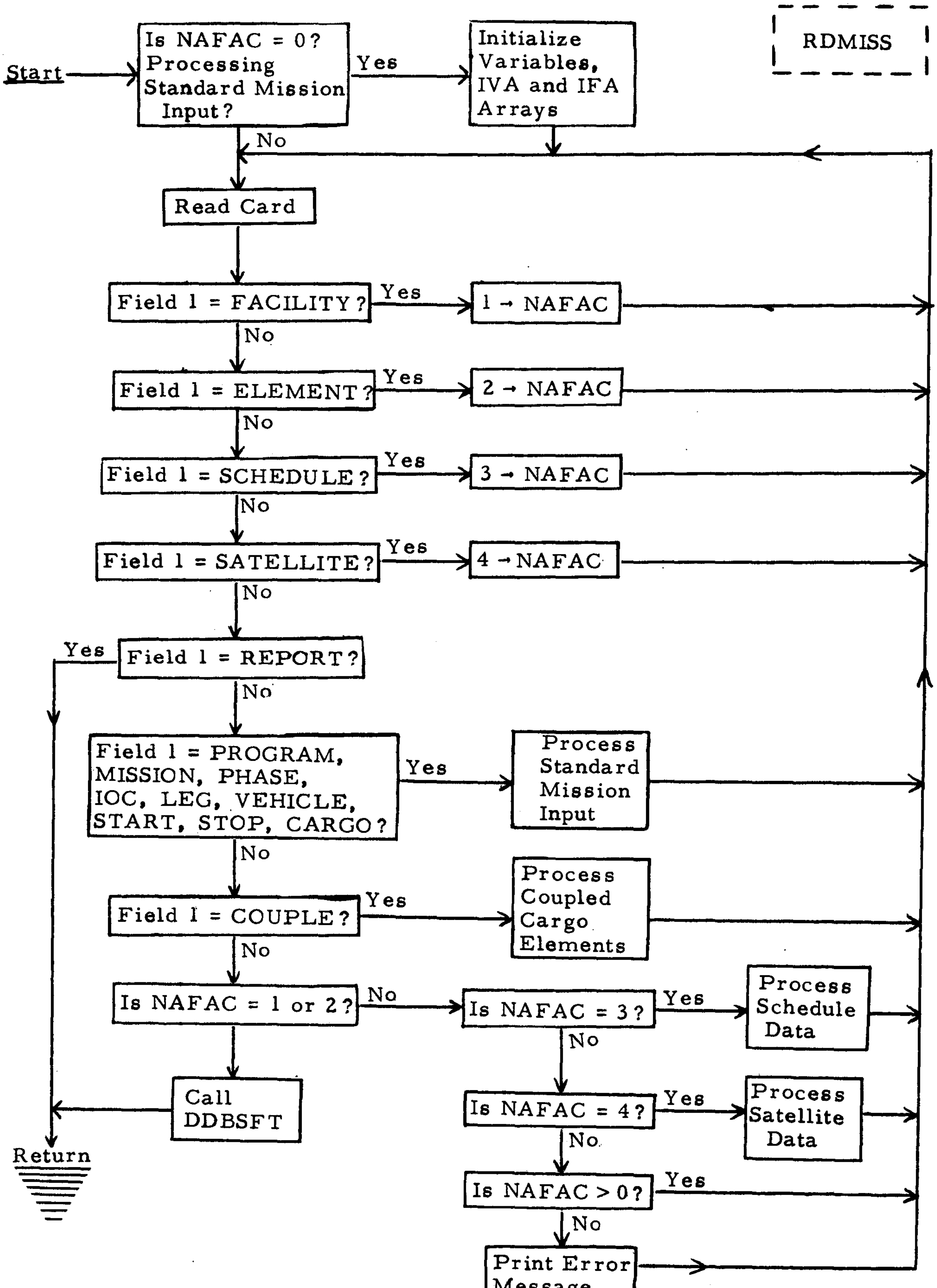
A schedule card contains the word SCHEDULE in field 1 and up to 7 IOC dates (numeric) in the remaining 7 fields. Additional cards may be used to define additional dates; these cards must be blank in field 1. The maximum number of dates is 20. On succeeding cards are the names of cargo elements in field 1 with the number of launches in each year in the fields corresponding to those years. Example:

SCHEDULE	1980	1985	1990
NAS-1CR	1	2	1
NAS-14ACR	2	3	2

Satellite data begins with the word SATELLITE in field 1 and the name of a cargo element in field 2. The following cards have the format:

Field	Contents
1	Year (4 digits)
2	Number of launches
3	Mode of delivery (DEPLOY, RETRIEVE, SERVICE)
4	Vehicle name
5	Deployment restriction (single if SINGLE, multiple if blank)

RDMISS sets NAFAC = 3 during schedule processing, NAFAC = 4 during satellite processing. RDMISS continues to read cards and assumes they are more schedule or satellite data provided field 1 does not contain one of the recognized key words. The data is transformed into entries into the level I cargo table.


RDMISS
Start
Is NAFAC = 0? Processing Standard Mission Input?
Yes
Initialize Variables, IVA and IFA Arrays
No
Read Card
Field 1 = FACILITY?
Yes
1 → NAFAC
No
Field 1 = ELEMENT?
Yes
2 → NAFAC
No
Field 1 = SCHEDULE?
Yes
3 → NAFAC
No
Field 1 = SATELLITE?
Yes
4 → NAFAC
No
Yes
Field 1 = REPORT?
No
Field 1 = PROGRAM, MISSION, PHASE, IOC, LEG, VEHICLE, START, STOP, CARGO?
Yes
Process Standard Mission Input
No
Field 1 = COUPLE?
Yes
Process Coupled Cargo Elements
No
Is NAFAC = 1 or 2?
No
Is NAFAC = 3?
Yes
Process Schedule Data
No
Call DDBSFT
Is NAFAC = 4?
Yes
Process Satellite Data
No
Return
Is NAFAC > 0?
Yes
No
Print Error Message

SECTION 27

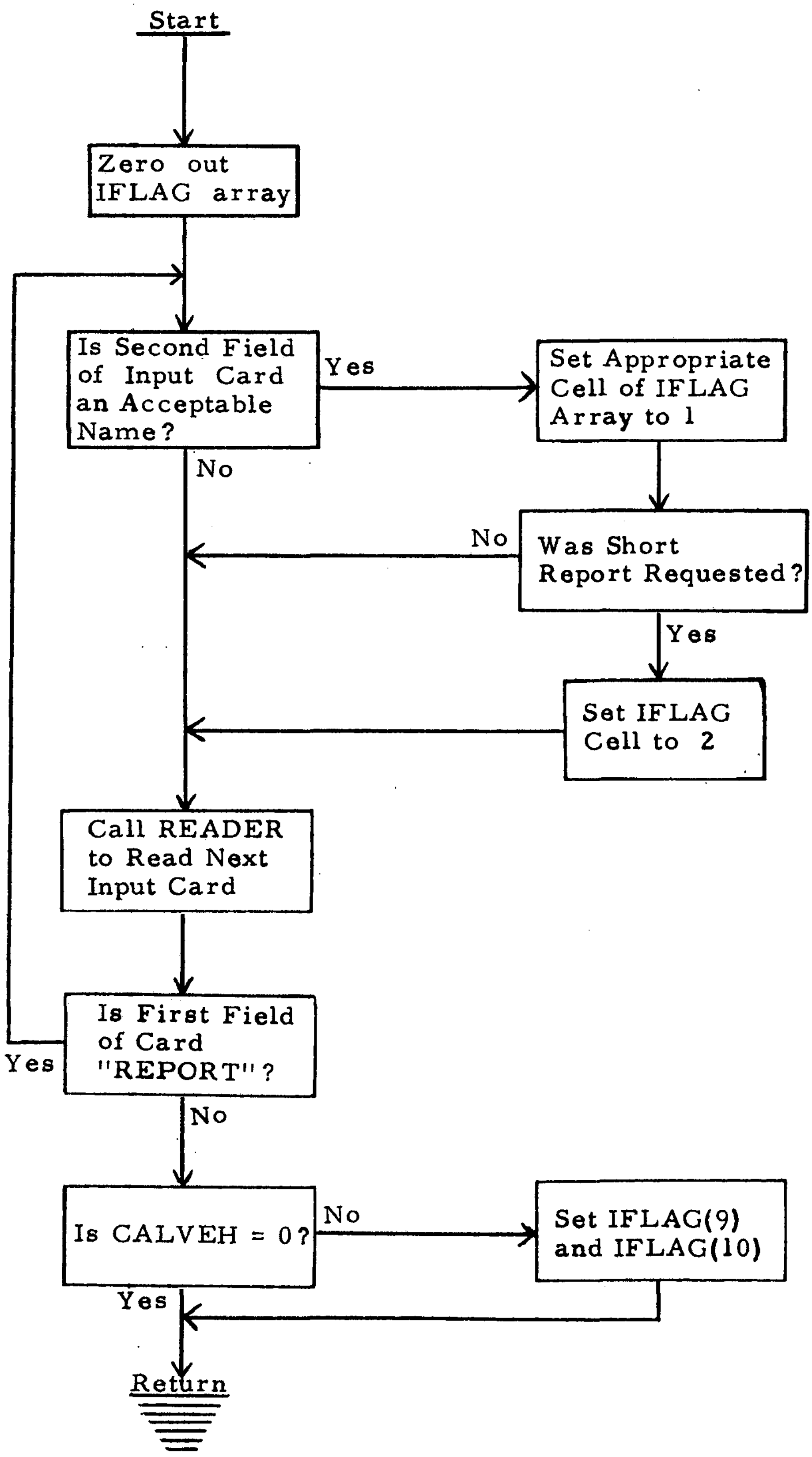
SUBROUTINE RDRPT

RDRPT reads the input REPORT request data cards and sets flags for later processing.

Array IFLAG is a list of flags which indicate which reports have been requested. Initially all flags are zeroed out. As each data card is read requesting a specific report, the appropriate flag is set to 1 (or 2 for a short report). The card format, as stated in the user's manual, is the word REPORT in field 1, the name of the report in field 2, and possibly the word SHORT in field 3. The reports which can be accommodated at present are:

Flag #	Report Name	Explanation	Routines which provide report
1	SPRINT	Summary of cargo traffic, giving the name, up weight, down weight, and load factor for each cargo element, arranged by program, mission, leg, vehicle, year and flight number	SPRINT
2	CONTAINER	Summary of container usage by year, leg, and name of cargo element	CNTRPT
3	FACILITY	Report on facility acquisitions in each year, arranged by program, mission and cargo element	FACRPT
4	TRAFFIC	Prints vehicle traffic tables	TRAFIC
5	VEHICLE	Vehicle acquisitions and utilization in terms of load factors	VEHRPT
6	COST	Cost reports	CSTRPT DPAGER
7	TABLES	Print input container, leg, spread, vehicle, facility, cost, and cargo element tables	TABLES
8	DEBUG	Intermediate printouts during leg processing for debugging purposes	TABLES LEGPRO TRAFIC

Flag #	Report Name		Routine
9	CALVEH	Calculate fleet vehicles as cargo	TRAFIC
10	PRTCAL	Intermediate printout during table calculations (LEGPRO temporarily stores the value of IFLAG(10) in IFLAG(4).)	TRAFIC
11	COST80	Same as IFLAG(6) - cost report - but using an 8 1/2 x 11 inch page format. COST80 also causes IFLAG(6) to be set to 1.	CSTRPT


Start
Zero out IFLAG array
Is Second Field of Input Card an Acceptable Name?
Yes
Set Appropriate Cell of IFLAG Array to 1
No
Was Short Report Requested?
No
Yes
Set IFLAG Cell to 2
Call READER to Read Next Input Card
Is First Field of Card "REPORT"?
Yes
No
Is CALVEH = 0?
No
Set IFLAG(9) and IFLAG(10)
Yes
Return

SECTION 28
SUBROUTINE RDSPD

RDSPD reads and stores input spread functions.

The coding is straightforward. Data cards consist of 8 fields of 10 characters each, and each 10-character field is read and stored in a pair of cells in A6, A4 format. The entries which are to represent numeric values (everything but the spread function name) are then converted into floating point numbers by subroutine VALUE.

The format specifications are given in the user's manual. RDSPD will call READER to read one input card at a time. For each card there are three possible actions:

1. Field 1 contains the word "TABLE." This means the end of the spread data, and control is returned to the calling routine (INPRO).
2. Field 1 contains a word other than "TABLE." This word is taken as the name of a new spread function, and the data in fields 2-8 define the spread function.
3. Field 1 is blank. This card is assumed to contain more data for the spread function last defined.

The first data card for each spread function contains the function name, the number of years, the first year, and up to 5 spread factors, representing percentages of cost to be applied in each year. The second and subsequent cards for each function, if any, contain up to 7 more spread factors. If any spread factors are left blank, the routine will ignore that field and all succeeding fields on that card. Thus, leaving a field blank is <u>not</u> equivalent to entering a zero.

All data are stored in floating point form except the name. The spread factors, which were input as percentages, are divided by 100 and then stored. The routine checks to see that the sum of the factors totals 1.0 -- or, allowing for truncation error, at least falls in the range 1.0 ± 0.01. Each spread function is stored in DDB as a group of variable length in the following order:

1. Number of years
2. First year
3. Spread factor 1
4. Spread factor 2
5. Spread factor 3

.
.
.

Immediately after the last spread factor of one spread function in DDB starts the data for the next function. Since the data is of variable length, matrix TBSPD is created to maintain pointers to each function:

TBSPD(1, J) = name (first 6 characters in Hollerith) of function #J

TBSPD(2, J) = name (last 4 characters)

TBSPD(3, J) = location in DDB of first cell of data for function #J (stored in floating point)

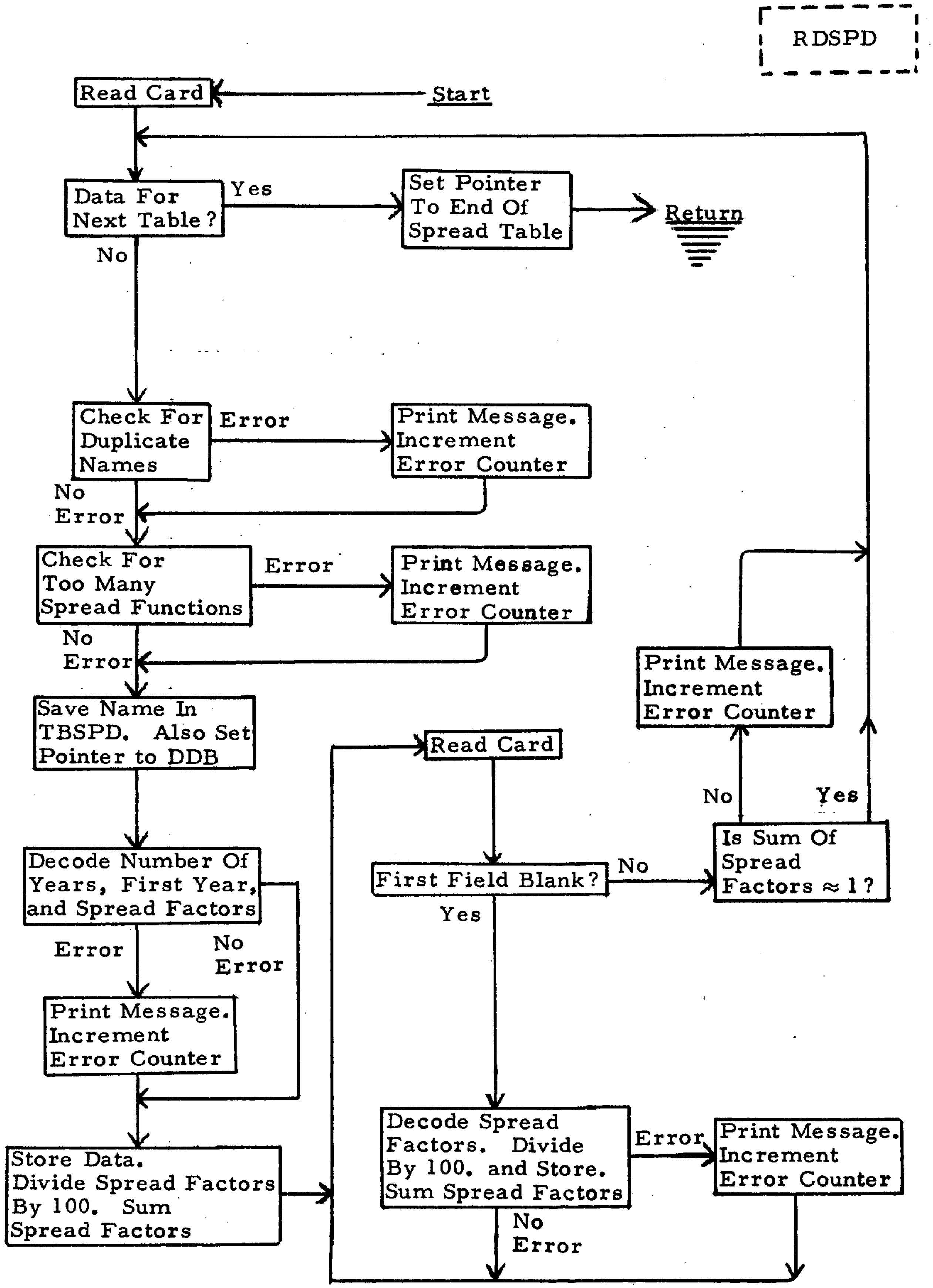
TBSPD is defined as interger within one subroutine. However, the two name words are Hollerith and the location entry is stored in floating point by using an equivalence between IX and XX.

At most 20 functions can be accommodated in TBSPD. Some other variables of interest are:

NSPD	Number of spread functions input
NBSPD	Location in DDB of first cell of first function
NLSPD	Location in DDB of last cell of last spread function
SPD	Sum of spread factors for current function; should be within the range (.99, 1.01)
ICPDDB	Counter for current position in DDB

The input errors detected by the routine are as follows:

1. Too many spread functions input (current limit is 20).
2. Duplicate function names.
3. Invalid value for one of the numeric entries.
4. Sum of spread factors does not total 100%.


RDSPD
Read Card
Start
Data For Next Table?
Yes
Set Pointer To End Of Spread Table
Return
No
Check For Duplicate Names
Error
Print Message. Increment Error Counter
No Error
Check For Too Many Spread Functions
Error
Print Message. Increment Error Counter
No Error
Save Name In TBSPD. Also Set Pointer to DDB
Decode Number Of Years, First Year, and Spread Factors
Error
No Error
Print Message. Increment Error Counter
Store Data. Divide Spread Factors By 100. Sum Spread Factors
Read Card
First Field Blank?
No
Yes
Is Sum Of Spread Factors ≈ 1?
No
Yes
Print Message. Increment Error Counter
Decode Spread Factors. Divide By 100. and Store. Sum Spread Factors
Error
Print Message. Increment Error Counter
No Error

SECTION 29

SUBROUTINE RDVEH

RDVEH reads and stores vehicle data and completes processing the leg table.

DATA CARDS

RDVEH calls subroutine READER to perform the actual card reading. If field 1 of the input card contains either of the words "TABLE" or "PROGRAM", then the vehicle data is considered to be finished, and RDVEH skips to statement 350 to complete processing of the leg table. Otherwise, the contents of the card are considered to be vehicle data. The data for each vehicle consists of the contents of three or more cards:

1. One card containing some basic data (required).

Field	Contents
1	Name
2	Propellant weight
3	Maximum number of flights/year
4	Maximum flights/lifetime
5	Lifetime in years
6	Minimum load (not used by program)
7	Volume limit
8	Deployment limit (max occupancy)

2. A second card with more basic data (required).

1	Blank
2	Development cost
3	Name of spread function for development costs
4	Production costs
5	Name of spread function for production costs
6	Operations costs per flight
7	Refurbishment costs (not used by program)
8	Name of propellant tank.

C

3. One card for each leg which the vehicle travels (required).

Field	Contents
1	Blank
2	Leg name
3	UPMAX (maximum weight vehicle can carry upwards if it returns empty)
4	DNMAX (maximum weight vehicle can carry downwards if it travels upwards empty)
5	EXPMAX (maximum weight vehicle can carry upwards if it does not return)

4. Stages cards for wet stages (propulsion units) and/or dry stages (structural units) - optional.

Field	Contents
1	For wet stages: word WET in columns 7-9. For dry stages: blank or word DRY in columns 7-9. Columns 1-6 and 10 must be blank.
2	Word STAGES
3	Name of vehicle which is stage 1.
4	Name of vehicle which is stage 2, if any.
5	Stage 3, if any.
6	Stage 4
7	Stage 5
8	Stage 6

5. ISP card (optional). Required if UPMAX, DNMAX and/or EXPMAX for any leg are not input but are to be calculated by DORCA.

1	The word ISP in columns 7-9. Columns 1-6 and 10 must be blank.
2	ISP number (specific impulse)
3	WSD (dry structure weight)
4	WNUP (non-usage propellant weight)
5	WINT (interstage weight)
6	WPBO (boil-off weight)
7	WNIE (non-impulsive propellant weight)
8	WACP (attitude-control propellant weight)

The first data card for each vehicle contains the vehicle name in field 1 (columns 1-10). Thereafter, every data card for this vehicle is blank in columns 1-6. The presence of non-blank characters in columns 1-6 is thus taken as a signal that a new vehicle is being defined. The minimum amount of data required for each vehicle is the two basic data cards plus at least one leg card. Variable JX is set equal to 1 as soon as the minimum has been reached (zero before); failure to input at least the minimum causes an error message to be printed.

DATA STORAGE

Most of the vehicle data is stored in the DDB array. For each vehicle, the variables are ordered in DDB as follows:

Location	Contents
L, L + 1	Vehicle name (A6, A4 format)
L + 2	Propellant
3	Max flights/year
4	Lifetime in flights
5	Lifetime in years
6	Minimum load
7	Nonrecurring development cost
8	Pointer to spread table for development cost
9	Propellant tank index
10	Recurring production cost
11	Pointer to spread table for production cost
12	Deployment limit (maximum occupancy)
13	Flight operations cost
14	Refurbishment cost
15	Volume constraint

Location	Contents
L + 16	ISP (Hollerith)
17	ISP number (specific impulse)
18	WSD (dry structure weight)
19	WNUP (non-usage propellant weight)
20	WPMAX (maximum propellant weight)
21	WINT (interstage weight)
22	WPBO (boil-off weight)
23	WNIE (non-impulsive propellant weight)
24	WACP (attitude-control propellant weight)
25	ISP number again
L + 26	Leg name (first 6 characters - A6 format)
27	Leg name (last 4 characters - A4 format)
28	UPMAX
29	DNMAX
30	EXPMAX
⋮	

Following L + 30 are additional groups of 5 words, one for each leg on which the vehicle flies, in the same format. The ISP data (items L + 16 through L + 25) is optional; if omitted, no space is saved for it. The index L indicates the location in DDB at which this information starts.

Additional information is stored in the vehicle table TBVEH (5, 30). For vehicle #J, five words are maintained in TBVEH(I, J), I = 1, ..., 5:

I	TBVEH(I, J)
1-2	Vehicle name (A6, A4 format)
3	Packed word Bits 0-17 contain length N (number of cells) of data for vehicle J; Bits 18-35 contain the location L of the start of the data in DDB.

I	TBVEH(I, J)
4	Packed word containing the dry stages used by this vehicle, if any. Up to 6 stages, 6 bits/stage.
5	Packed word containing the wet stages used by this vehicle, if any. Up to 6 stages, 6 bits/stage.

PERFORMANCE CALCULATIONS

For each leg the vehicle flies, DORCA II must have available the performance quantities UPMAX, DNMAX and EXPMAX. These may be input by the user or computed by DORCA II. If their respective fields on the input card (fields 3, 4 and 5) are blank, DORCA II will attempt to compute these quantities. In order to compute them, DORCA II needs three sets of data: (1) A WET STAGES card for this vehicle, (2) ISP cards for each of the vehicles corresponding to the wet stages for this vehicle, and (3) the velocity increment Δ V for this leg, obtained from TBLEG. Subroutine PERLNK is called to compute the performance values.

POST PROCESSING

The few statements from #320 to #350 wrap up the indexing and possibly error printing for this vehicle. Following statement 350, RDVEH finishes processing of the leg table, filling in indices where entries in the leg table refer to vehicle names which were not available at that time. Also, the routine check that every leg name entered on a leg card input as part of the vehicle data corresponds to an entry in the leg table. At the conclusion of RDVEH, the leg table has the following format: the 12 elements of column J contain the following data on leg #J:

Word	Format	Contents
1-2	A6, A4	Leg name
3-4	A6, A4	Name of lower (previous) leg name
5	Real	Maximum occupancy
6	Packed	Default vehicles on this and lower legs
7	Integer	Previous leg number

Word	Format	Contents
8	-	Not used
9	Real	Δ V
10	Real	Longshoring flag
11	Integer	Alternate vehicle number
12	-	(not used)

Word 6 for each leg has the following appearance:

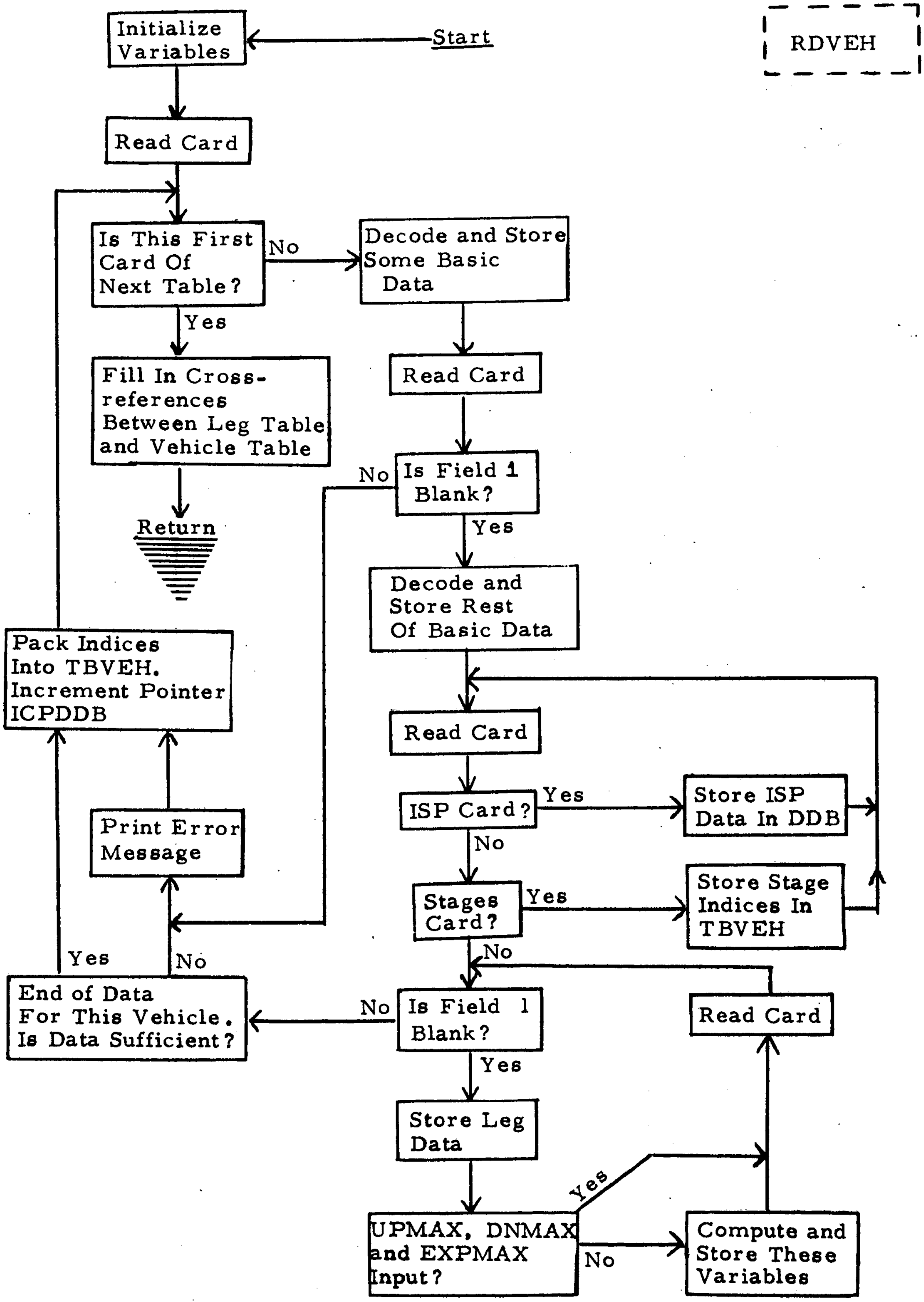
0	V_1	V_2	V_3	V_4
0	12	18	24	30

where V_1 is the index of the default vehicle for the current leg,

V_2 is the index of the default vehicle for the previous (lower) leg, if any;

V_3 is the index of the default vehicle for the second lower leg, if any;

V_4 is the index of the default vehicle for the third lower leg, if any.


RDVEH
Start
Initialize Variables
Read Card
Is This First Card Of Next Table?
No
Yes
Decode and Store Some Basic Data
Read Card
Fill In Cross-references Between Leg Table and Vehicle Table
Return
Is Field 1 Blank?
No
Yes
Decode and Store Rest Of Basic Data
Pack Indices Into TBVEH. Increment Pointer ICPDDB
Read Card
ISP Card?
Yes
Store ISP Data In DDB
No
Print Error Message
Stages Card?
Yes
Store Stage Indices In TBVEH
No
Yes
No
End of Data For This Vehicle. Is Data Sufficient?
No
Is Field 1 Blank?
Read Card
Yes
Store Leg Data
Yes
UPMAX, DNMAX and EXPMAX Input?
No
Compute and Store These Variables

SECTION 30

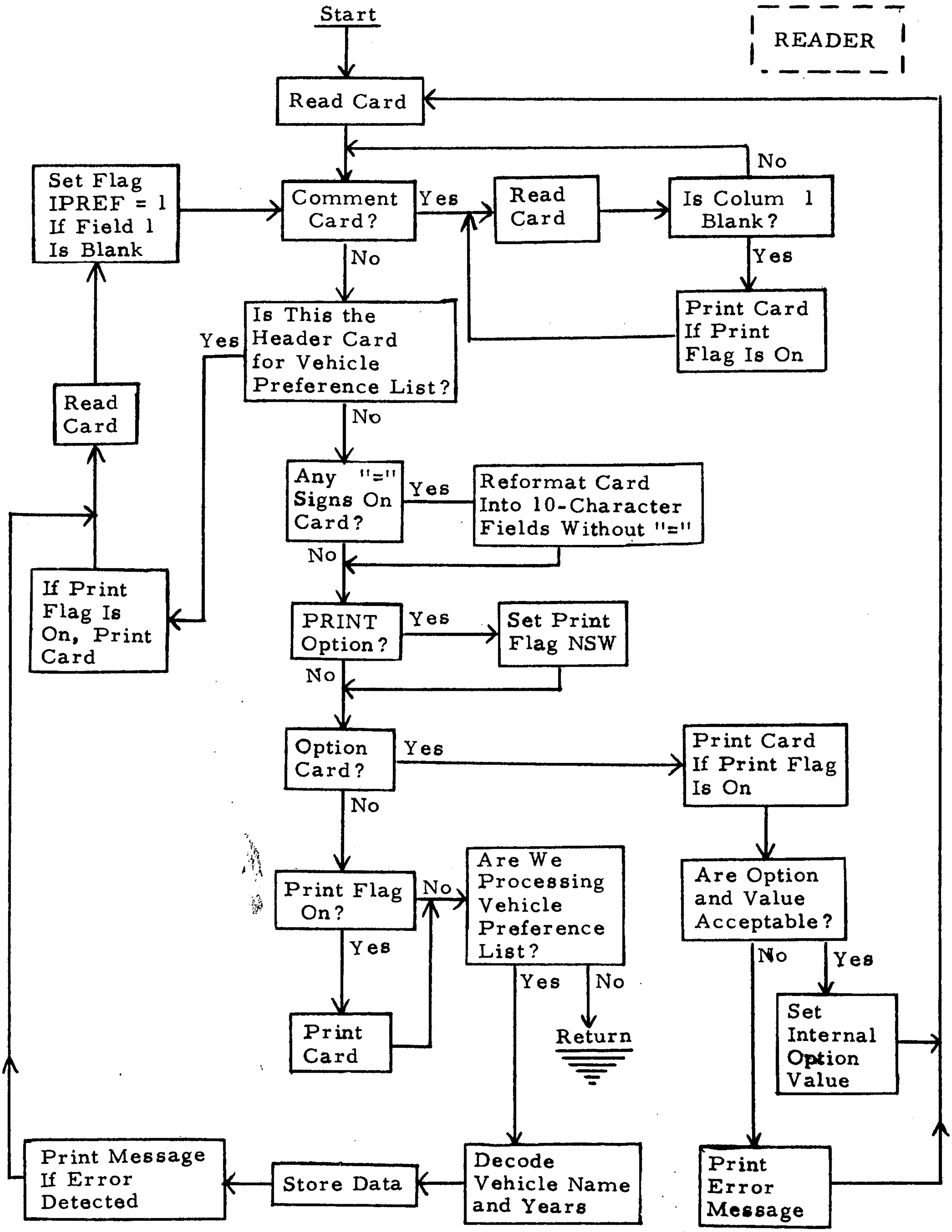
SUBROUTINE READER

READER is called from the various input routines to read a single card, and print it if the print option is currently on. READER will also process certain types of data cards: comment cards, option cards, and the vehicle preference list. Other types of cards are processed by the subroutines which call READER.

Certain types of cards (include OPTION and mission CARGO cards) have a free-field format which uses the equals symbol as a separator. READER searches the input card for the character "="; if any are found, the data preceding and following is reformatted into 10-character fields to simplify processing by the calling routines.

Option cards are processed by comparing the name and alphanumeric value on the input card with a list of acceptable names and values. If a match is made, the appropriate value is set in the KOPT array. Illegal names and values cause an error message.

If field 1 of an input card contains the word PREFERENCE, then the succeeding cards are taken to define the vehicle preference list until a card is read which is not blank in column 1. READER processes the entire preference list and packs the information into array VPREF


READER
Start
Read Card
Set Flag IPREF = 1 If Field 1 Is Blank
Comment Card?
Yes
Read Card
Is Colum 1 Blank?
No
Yes
Print Card If Print Flag Is On
No
Is This the Header Card for Vehicle Preference List?
Yes
Read Card
If Print Flag Is On, Print Card
No
Any "=" Signs On Card?
Yes
Reformat Card Into 10-Character Fields Without "="
No
PRINT Option?
Yes
Set Print Flag NSW
No
Option Card?
Yes
Print Card If Print Flag Is On
Are Option and Value Acceptable?
No
Yes
Set Internal Option Value
Print Error Message
No
Print Flag On?
No
Are We Processing Vehicle Preference List?
Yes
No
Return
Yes
Print Card
Decode Vehicle Name and Years
Store Data
Print Message If Error Detected

SECTION 31

SUBROUTINE REPORT

REPORT calls subroutines to generate the reports requested by input. These reports can be any or all of the following list:

Report Number	Report Name	Explanation
1	SPRINT	Cargo traffic
2	CONTAINER	Container usage
3	FACILITY	Facility acquisition
4	TRAFFIC	Vehicle traffic
5	VEHICLE	Vehicle acquisitions and utilization
6	COST	Cost report

If any intermediate or debugging printout was requested, that information is printed via a call to subroutine TABLES preceding the formal reports.

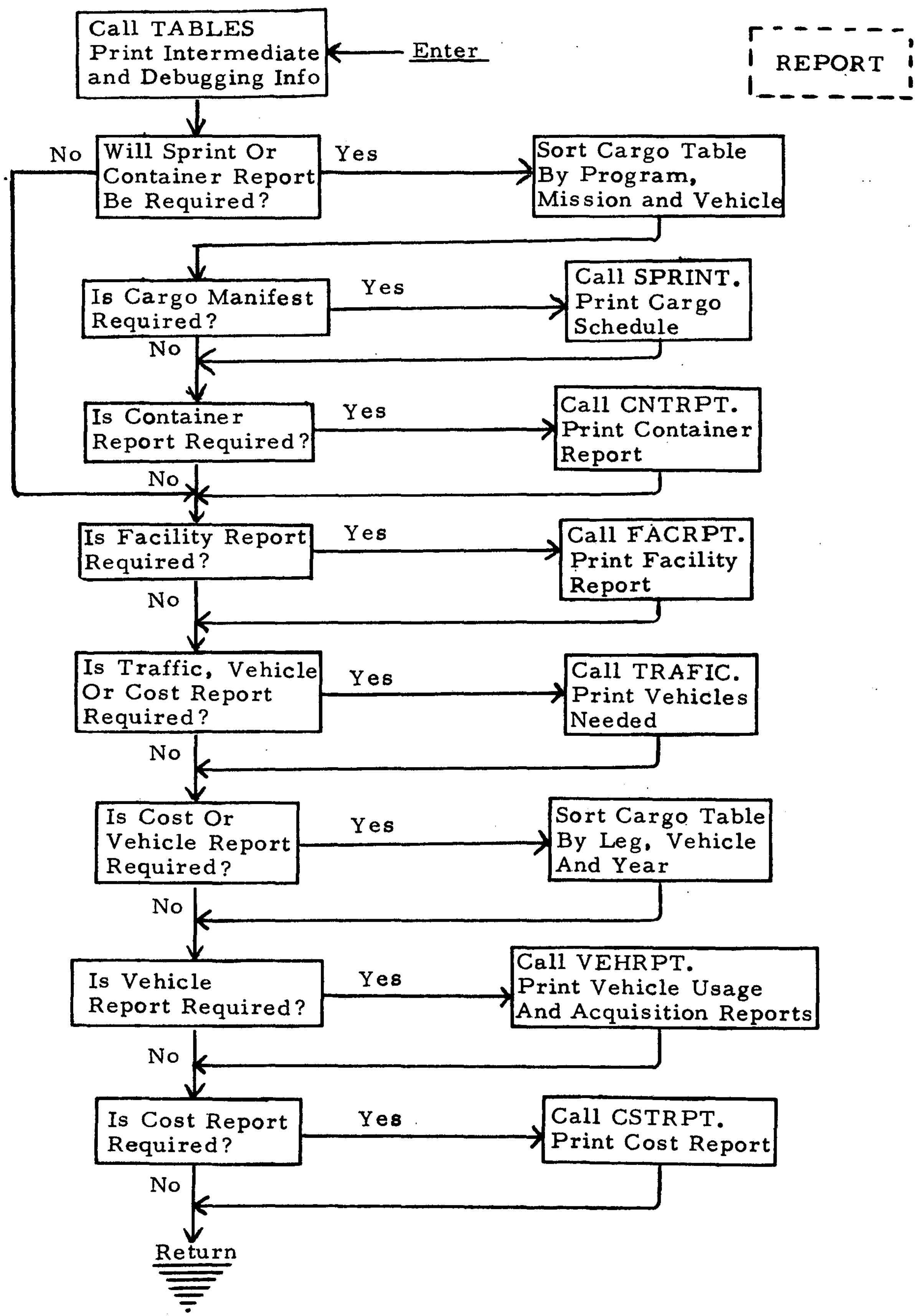
If either report #1 or report #2 or both have been requested, the level II cargo tables are first sorted by leg, vehicle, year and flight number, in that order. If reports 4, 5, and/or 6 were requested, subroutine TRAFIC is called to determine the actual number of vehicles needed. If report 5 and/or 6 was requested, the cargo tables are first sorted by program, mission and vehicle. Since the cargo tables are stored on external files, subroutine MERGE is called to perform the out-of-core sort of the cargo table matrix on element #KEL:

KEL = 1: sort on leg/vehicle/year/flight no.

KEL = 2: sort on program/mission/vehicle

Since the program is segmented in OVERLAY format, the names of the subroutines called from REPORT are somewhat disguised:

Segment Name	Subroutine Name
TABLES	TABLES
SEG32	MERGE
SEG33	SPRINT
SEG36	TRAFIC
CSTRPT	CSTRPT, SPDAP, DPAGER


REPORT
Enter
Call TABLES
Print Intermediate
and Debugging Info
Will Sprint Or
Container Report
Be Required?
No
Yes
Sort Cargo Table
By Program,
Mission and Vehicle
Is Cargo Manifest
Required?
Yes
Call SPRINT.
Print Cargo
Schedule
No
Is Container
Report Required?
Yes
Call CNTRPT.
Print Container
Report
No
Is Facility Report
Required?
Yes
Call FACRPT.
Print Facility
Report
No
Is Traffic, Vehicle
Or Cost Report
Required?
Yes
Call TRAFIC.
Print Vehicles
Needed
No
Is Cost Or
Vehicle Report
Required?
Yes
Sort Cargo Table
By Leg, Vehicle
And Year
No
Is Vehicle
Report Required?
Yes
Call VEHRPT.
Print Vehicle Usage
And Acquisition Reports
No
Is Cost Report
Required?
Yes
Call CSTRPT.
Print Cost Report
No
Return

SECTION 32

PROGRAMS SEG22, SEG32, SEG33, SEG36

These tiny programs are linkages in the OVERLAY structure.

SEG22 calls subroutine TRAFIC as part of the leg processing. The arguments are MAXVEH, the maximum number of vehicles which can be accomodated in the DVTT matrix; and the beginning of an available area of core which can be used for DVTT.

SEG36 calls TRAFIC as part of the REPORT generation. The argument representing the buffer to be used for DVTT is a different area from that is SEG22.

SEG32 calls MERGE to perform an out-of-core sort of the Level 2 cargo table on element #KEL. The program computes NFILE, the amount of free space available in array DDB to be used for sorting and merging, expressed as a multiple of 510-word groups. This free space corresponds to the matrix FILE in subroutine MERGE. If NFILE ≥ 3, the free space in DDB is used for FILE; if NFILE < 3, an array A of 3 x 510 words in labelled common/ASDAT/ is used for FILE.

SEG33 calls SPRINT to print the cargo manifest and/or load factors. The two arguments in the calling sequence are the same to permit SPRINT to refer to the major matrix by a real name (XLF) as well as an integer name (LF).

SECTION 33
SUBROUTINE SORT

Given a matrix of M rows and N columns, this routine rapidly sorts the columns such that element number L of each column forms an increasing sequence. Upon option, the sort is either alphanumeric or algebraic.

The Lth elements should all be either normalized floating point or integers but not mixed. The format of the other elements is irrelevant and can be mixed.

CALL SORT (A, M, N, L)

where A is the M×N matrix

M is the number of rows of A

N is the number of columns of A

L is the number of the element in each column on which the sort is computed, i. e., the number of the row whose elements are to be sorted into increasing order; the other elements in each column are simply permuted in the same manner.

For simplification and without loss of generality, assume the matrix A has only one row, so that the algorithm amounts to sorting the elements of an array A_i ($i = 1, \ldots, N$).

Step 1. Initialization. Let I = 1, J = N, M = 1. I and J denote the boundaries of an interval within the range of the array. M is a pointer to the lists IL and IU, which are used to contain the boundaries of other intervals to be scrutinized later.

Step 2. If $I \geq J$, to go step 7. Otherwise, set K to the value of I, L to the value of J, and IJ to the midpoint of the interval [I, J]. Interchange elements A_I, A_J and A_{IJ} as necessary until $A_I \leq A_{IJ} \leq A_J$. The boundary and midpoints are now in order.

Step 3. Decrement L by 1 until we find an $A_L \leq A_{IJ}$. Increment K by 1 until we find an $A_K \geq A_{IJ}$.

Step 4. If $K \leq J$, interchange elements A_K and A_L and return to step 3. If $K > L$, a crossover has occurred, and the algorithm now divides the interval [I, J] into two parts, [I, L] and [K, J]. Choose the longer of these two parts and save the boundaries in the arrays IL and IU at position M for later scrutiny, then increment M by 1, Reset I and J to the boundaries of the shorter part.

Step 5. If the interval [I, J] is large, containing more than 10 elements, return to step 2. If [I, J] is small, with 10 or fewer elements, go to step 6.

Step 6. Neighbor interchange. This is efficient for small intervals [I, J] only. For K = I, I-1, I-2, ... , interchange A_K and A_{K+1} until either K = 0 or $A_K \leq A_{K+1}$ already. Now increment I by 1, and repeat this procedure until I = J, then go to step 7.

Step 7. At this point, interval [I, J] is ordered, and it is necessary to determine whether any other intervals within the range [1, N] remain unordered. Decrease M by 1. If now M = 0, there remain no unordered intervals, and the algorithm is finished. If M > 0, however, there remain M intervals which were saved for further scrutiny. Retrieve the values of I and J stored in arrays IL and IU at position M and to to step 5.

Sort mode. According to the option variable MODE in labeled common the sort can be either algebraic (MODE $\neq$ 0) or alphanumeric (MODE = 0). In the algebraic mode, negative numbers are considered to be less than positive ones. In the alphanumeric mode, the values being sorted are assumed to represent not pure numbers but rather alphabetic or other alphanumeric characters; in this case the sign bit is not taken as a plus or minus sign but rather as the highest order bits, so that words which register as negative numbers in the computer are actually large "positive" characters. In this mode, the sorted matrix contains these "negative" numbers at the end of the matrix. For example, the sequence

3 6 2 -1 1 -4 3 -2

will be sorted into the sequence

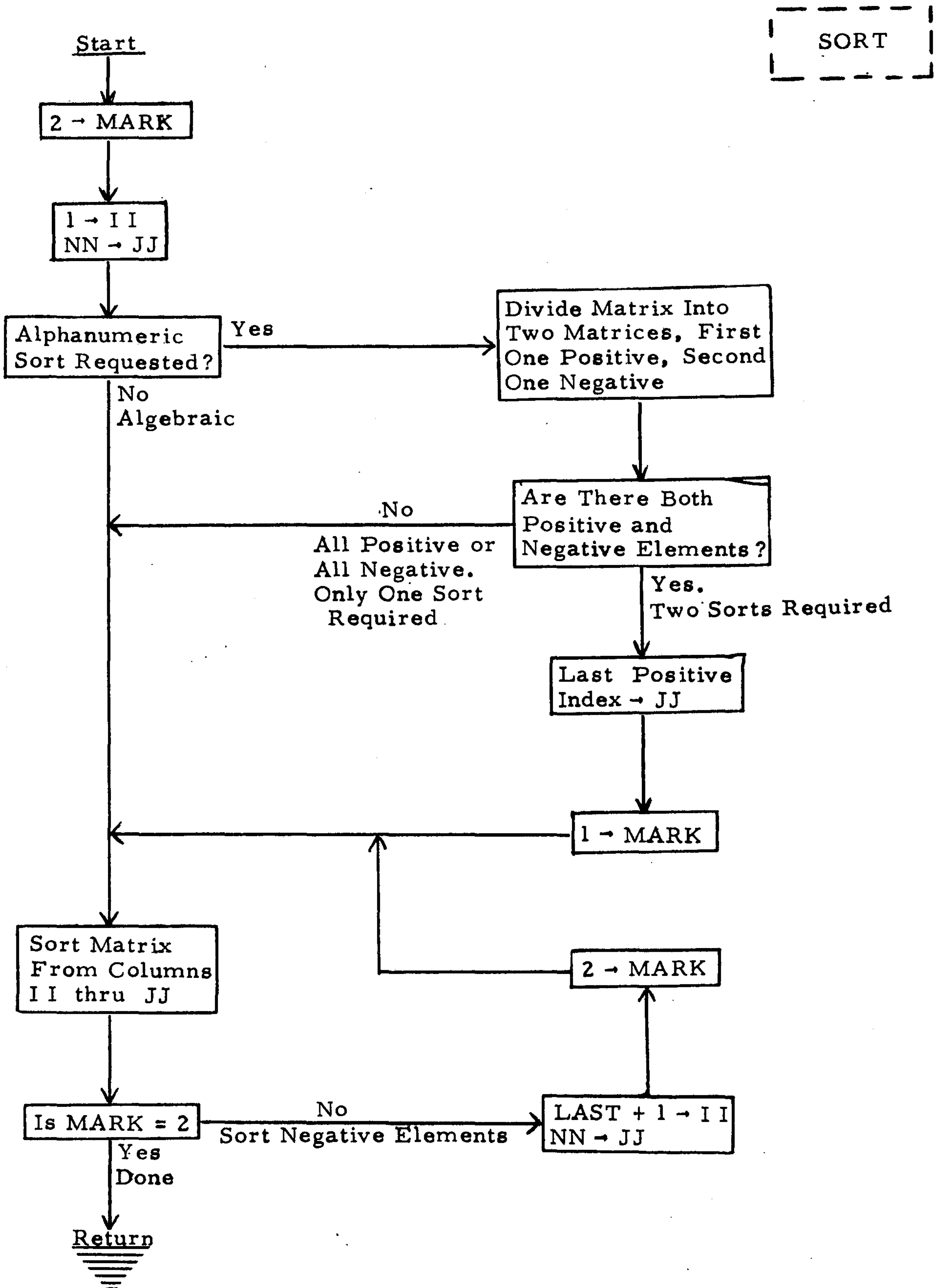
-4	-2	-1	1	2	3	3	6	(algebraic mode)
1	2	3	3	6	-4	-2	-1	(alphanumeric mode)

The method for handling the alphanumeric mode is to first divide the matrix into two matrices, one containing only positive numbers, the other only negative numbers, then sorting each part separately. The separating operation plus two smaller sorts takes about the same time as a single sort of the entire matrix.

In the FORTRAN code, integer arithmetic is used for all operations (by defining A and all other quantities as integers) because it is faster than floating point arithmetic. The subroutine will work properly if the elements of A are either all integers or all normalized floating point numbers. However, care should be exercised if the elements are Hollerith characters, since some characters can cause the elements to be treated as negative numbers.

Execution time of the algorithm is proportional to N$\cdot$log (N).

The subroutine was developed by Richard C. Singleton of the Stanford Research Institute and modified for DORCA.


SORT
Start
2 → MARK
1 → I I
NN → JJ
Alphanumeric Sort Requested?
Yes
Divide Matrix Into Two Matrices, First One Positive, Second One Negative
No
Algebraic
Are There Both Positive and Negative Elements?
No
All Positive or All Negative. Only One Sort Required
Yes.
Two Sorts Required
Last Positive Index → JJ
1 → MARK
Sort Matrix From Columns I I thru JJ
2 → MARK
Is MARK = 2
No
Sort Negative Elements
LAST + 1 → I I
NN → JJ
Yes
Done
Return

SECTION 34
SUBROUTINE SPDAP

The name SPDAP comes from SPreads Development And Production costs. Given the name of a vehicle or facility, and a count of units acquired in each year, and a spread function, SPDAP produces a list of costs incurred in each year by applying the spread function.

A spread function defines the manner in which the cost of a unit is to be spread over several years. Each spread function input to DORCA consists of three parts:

1. The span of N years over which the cost is to be spread. (N is denoted IYRSP in this routine.)
2. A list of spread factors f_i for each of the N years indicating the portion of the cost to be paid in each year. The sum of these factors must be 1.0 (100%).
3. A parameter M indicating that the unit is actually delivered in the Mth year of the N-year span. If, for example, N = 10 and M = 5, the unit is paid for over 10 years, starting 4 years before it is actually delivered. (M is denoted by the variable IYRIOC in SPDAP.)

Thus, for a unit purchased in a given year y, the cost is distributed as follows:

$$
\begin{array}{ll}
\$\ (\text{unit cost}) \times f_1 \text{ in year} & y-M+1 \\
\$\ (\text{unit cost}) \times f_2 \text{ in year} & y-M+2 \\
\quad\vdots & \\
\$\ (\text{unit cost}) \times f_N \text{ in year} & y-M+N
\end{array}
$$

When a number of units of a given vehicle or facility have been purchased over a period of years, the total cost incurred in a given year y may be expressed as

$$\text{Total cost} = (\text{unit cost}) \sum_{i=1}^{N} f_i \, C_{y+M-i}$$

where C_k denotes the number of units purchased in year k.

The argument list of SPDAP consists of five variables and arrays:

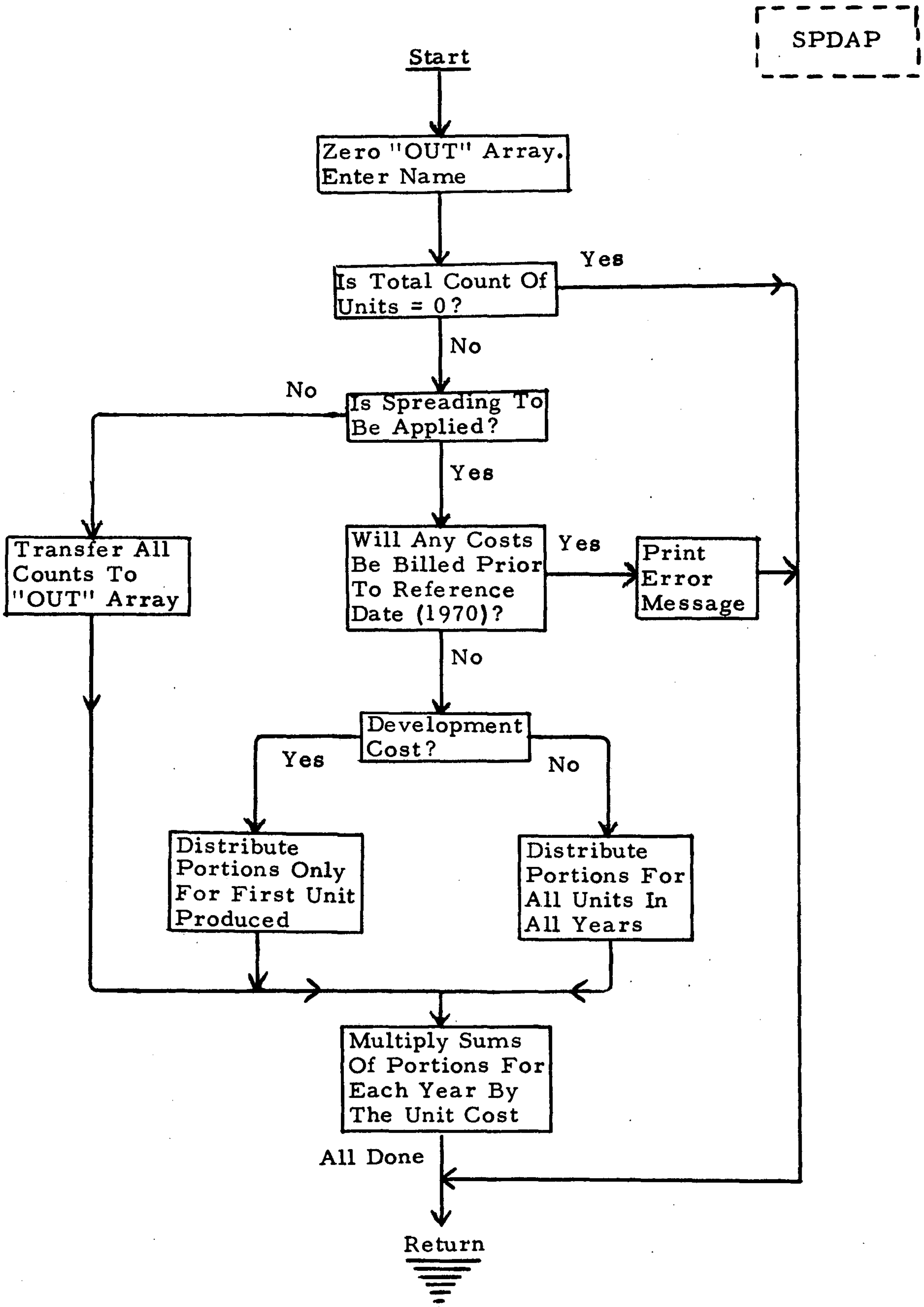
IN	An array in which words 1-2 contain the vehicle or family name, word 3 contains the total count of units acquired in all years, and words 4, 5, 6, etc., contain the number of units acquired in year 1, 2, 3, etc.
COST	is the unit cost, in millions of dollars.
SPREAD	is an array defining the spread function, in which word 1 is the number of years (IYRSP) which the function spans, word 2 is the relative year of delivery (IYRIOC), and words 3, 4, 5, etc., contain the spreading factors for the first, second, third, etc., year of the function. If IYRSP = 0 there is no spreading, i. e., costs are paid fully in the year in which incurred.
OUT	is the array to be calculated by SPDAP. The format is the same as array IN except that all unit counts have been replaced by costs.
IFLG	= 1 for nonrecurring development costs, = 0 for recurring production or operations costs.

Recurring (production or operations) costs and nonrecurring (development) costs are handled somewhat differently. Production costs are multiplied by the number of units and spread over the years as indicated above. For development, SPDAP assumes only one unit, in the first year indicated by a nonzero count in array IN. In other words, development costs are paid only for the first unit produced. If array IN contains a count greater than 1 or units purchased in more than one year when IFLG = 1, that data is simply ignored.

The computation has three basic steps:

1. Zero out the OUT array.
2. For year Y in which the number of units purchased (the entry in array IN) is nonzero, multiply the number of units by the spread factors and accumulate the products in array OUT in the slots correspond to years Y-M+1 through Y-M+N. If IYRSP = 0, there is no spreading and the data is simply transferred from array IN to OUT.
3. Multiply the sum for each year by the unit cost; also compute the total cost over all years.

The variable FYEAR denotes the first year that any cargo has been shipped, after subtracting the relative date 1970. The test at statement number 120 determines whether the spreading function for the first units in the IN array will cause any part of the cost to be billed for a year preceding 1970, which is illegal in DORCA. If so, an error message is printed and control is returned to the calling program (CSTRPT) with no processing. The presence of the variable L = FYEAR in statements 140 and 200 simply shifts the data if necessary so that the cost array always starts with the reference date 1970.


SPDAP
Start
Zero "OUT" Array. Enter Name
Is Total Count Of Units = 0?
Yes
No
Is Spreading To Be Applied?
No
Yes
Transfer All Counts To "OUT" Array
Will Any Costs Be Billed Prior To Reference Date (1970)?
Yes
Print Error Message
No
Development Cost?
Yes
No
Distribute Portions Only For First Unit Produced
Distribute Portions For All Units In All Years
Multiply Sums Of Portions For Each Year By The Unit Cost
All Done
Return

SECTION 35
SUBROUTINE SPRINT

SPRINT prints a summary of volume and load factors and a complete manifest of all cargo items shipped. This is an optional report which is generated if a REPORT SPRINT card was input to the report table. The cargo manifest is omitted if a short report was requested.

Before calling SPRINT, subroutine REPORT calls Subroutine SEG22-MERGE to sort the entire Phase II cargo table by leg, vehicle, year, and flight number. The first lines of print are a title "CARGO MANIFEST" and the column headings, which are:

PROGRAM	-	name of program with which cargo item is associated
MISSION	-	name of mission with which cargo item is associated
LEG	-	name of leg on which cargo item travels
VEHICLE	-	name of vehicle used to carry cargo item
YEAR	-	date of flight
FLIGHT	-	flight number to which cargo item is assigned
CARGO	-	name of cargo element
WT UP	-	weight of item if shipped upwards (zero if shipped down)
WT DOWN	-	weight of item if shipped downwards (zero if shipped up)
ELF	-	load factor of item (see below)

The load factor of a cargo item is the weight of the item (expressed as equivalent up-weight) as a proportion of the total weight carried by the vehicle round trip (also expressed as equivalent up-weight). It is computed by the following equation in subroutine LEGPRO:

$$\text{load factor} = \frac{\text{item wt} \cdot \text{factor}}{\text{tot wt up} + \text{tot wt down} \cdot \left(\frac{\text{UPMAX}}{\text{DNMAX}}\right)}$$

where item wt = weight of assigned cargo item (one direction only)
tot wt up = total weight carried upwards by vehicle on this flight
tot wt down = total weight carried downwards by vehicle on this flight

UPMAX = maximum vehicle capacity (lbs) upwards

DNMAX = maximum vehicle capacity (lbs) downwards

$$\text{factor} = \begin{cases} 1 & \text{if cargo item is traveling upwards} \\ \dfrac{\text{UPMAX}}{\text{DNMAX}} & \text{if cargo item is traveling downwards} \end{cases}$$

The sum of load factors for all cargo items on a single flight must be 1.

The procedure is straightforward. The routine processes each item in the cargo table in turn, extracting from each a composite index composed of the leg number, vehicle number, year, and flight number. Whenever the composite index of an item differs from that of the previous item, indicating a new flight, the sums of up weights, down weights and load factors from the previous flight are printed, then reset to zero. From each item the routine extracts the various indices and data to generate one line of print for the headings above. The up weights and down weights are obtained by multiplying the original input to the cargo element table by the bulk load factor BLF, where $0 < \text{BLF} \leq 1$. $\text{BLF} < 1$ only for bulk cargo which has been subdivided into two or more portions traveling in different containers.

The summaries of volume and load factors are printed for both the short and long report. Each summary gives the average volume/load factor for each vehicle and leg in each year, plus a "TOTAL" which is the average for all years. Subtotal averages are printed for each vehicle averaged over all legs, as well as a grand total average over all vehicles.

These factors are accumulated in the large XLF/LF matrix:

1	1	NYRS	1	NYRS	1	NYRS
I	T_L	Load factors	T_F	flights	T_V	Volume factors

JJ

3*NYRS+4

For NYRS = 30,

Column 1	:	Composite indexI = vehicle index x 100 + leg index
2	:	Total load factors for all years
3-32	:	Load factors for each year
33	:	Total number of flights for all years
34-63	:	Number of flights in each year
64	:	Total volume factors for all years
65-94	:	Volume factors for each year

Each row corresponds to a particular vehicle/leg combination. In addition, one row is reserved for each vehicle for subtotals by using the key leg number of 63. Finally, one line is reserved for the grand totals by using the Key index I = 3131.

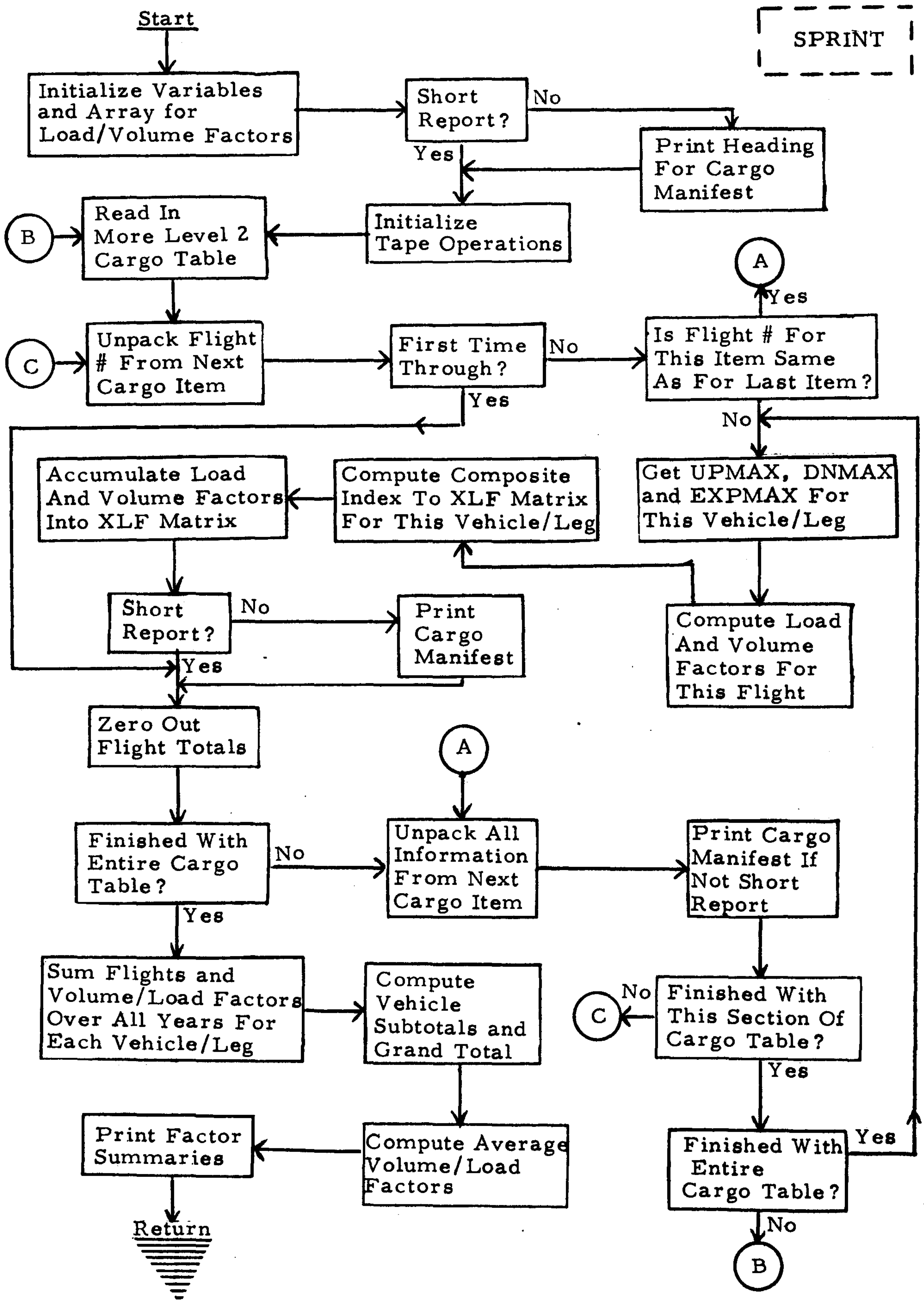
The load factor TRLF for a given flight is the ratio of the total weight shipped up (TWTUP) and down (TWTDW) on that flight to the capacity of the vehicle (UPMAX/EXPMAX and DNMAX:

$$TRLF = \begin{cases} \frac{TWTUP}{UPMAX} + \frac{TWTDW}{DNMAX} & \text{if vehicle returns} \\ \frac{TWTUP}{EXPMAX} & \text{if vehicle is expended} \end{cases}$$

The volume factor VOLF is the ratio of the total volume VOLUME of all cargo shipped on that flight (up-direction only) to the vehicle's volume capacity VØLMAX.

$$VØLF = \frac{VØLUME}{VØLMAX}$$

The load factors and volume factors are accumulated in XLF, as are the number of flights, for all items in the Level 2 cargo table. When the cargo table has been completely processed, the accumulated factors for each year are divided by the flights in that year, yielding average load and volume factors, which are printed.


SPRINT
Start
Initialize Variables and Array for Load/Volume Factors
Short Report?
No
Yes
Print Heading For Cargo Manifest
Initialize Tape Operations
B
Read In More Level 2 Cargo Table
A
Yes
C
Unpack Flight # From Next Cargo Item
First Time Through?
No
Is Flight # For This Item Same As For Last Item?
Yes
No
Accumulate Load And Volume Factors Into XLF Matrix
Compute Composite Index To XLF Matrix For This Vehicle/Leg
Get UPMAX, DNMAX and EXPMAX For This Vehicle/Leg
Short Report?
No
Print Cargo Manifest
Compute Load And Volume Factors For This Flight
Yes
Zero Out Flight Totals
A
Finished With Entire Cargo Table?
No
Unpack All Information From Next Cargo Item
Print Cargo Manifest If Not Short Report
Yes
Sum Flights and Volume/Load Factors Over All Years For Each Vehicle/Leg
Compute Vehicle Subtotals and Grand Total
No
C
Finished With This Section Of Cargo Table?
Yes
Print Factor Summaries
Compute Average Volume/Load Factors
Finished With Entire Cargo Table?
Yes
No
B
Return

SECTION 36

SUBROUTINE SWITCH

SWITCH is called from the RDCRG and RDMISS routines to retain all non-facility cargo elements and all activated cargo elements that are facilities.

The usage is

CALL SWITCH (N)

where N is a pointer to the Nth cargo element to be retained.

The variable LCE points to the last saved cargo element. SWITCH interchanges the Nth cargo element data with the LCE + 1^{th} cargo element. LCE is increased by 1. If the cargo element is a facility, then the facility table undergoes a switch based on the variable LFAC. The cargo element that was moved is updated for new facility pointer. After all mission data has been read in, RDMISS will move the cargo element table deleting unused facilities and move the level I cargo table deleting the unused cargo elements.

SECTION 37

SUBROUTINE TABLES

TABLES prints, upon request, internal tables for debugging purposes. This routine is called from subroutine REPORT.

If IFLAG(8) ≠ 0 (i.e., if the DEBUG report was requested in the input REPORT table), the following variables are printed:

1. Counters and limits of certain internal tables (but not the tables themselves):

NBSPD, NLSPD, NSPD	(spread data)
NBVEH, NLVEH, NVEH, NWVEH	(vehicle data)
NBFAC, NLFAC, NFAC, NWFAC	(facility data)
NBCE, NLCE, NCE, NWCE	(cargo elements)
NBMISS, NLMISS, LNGTH, NWMISS	(Level I cargo table)
NBDDB, NLDDB, NDDB, NWDDB	(Level II cargo table)

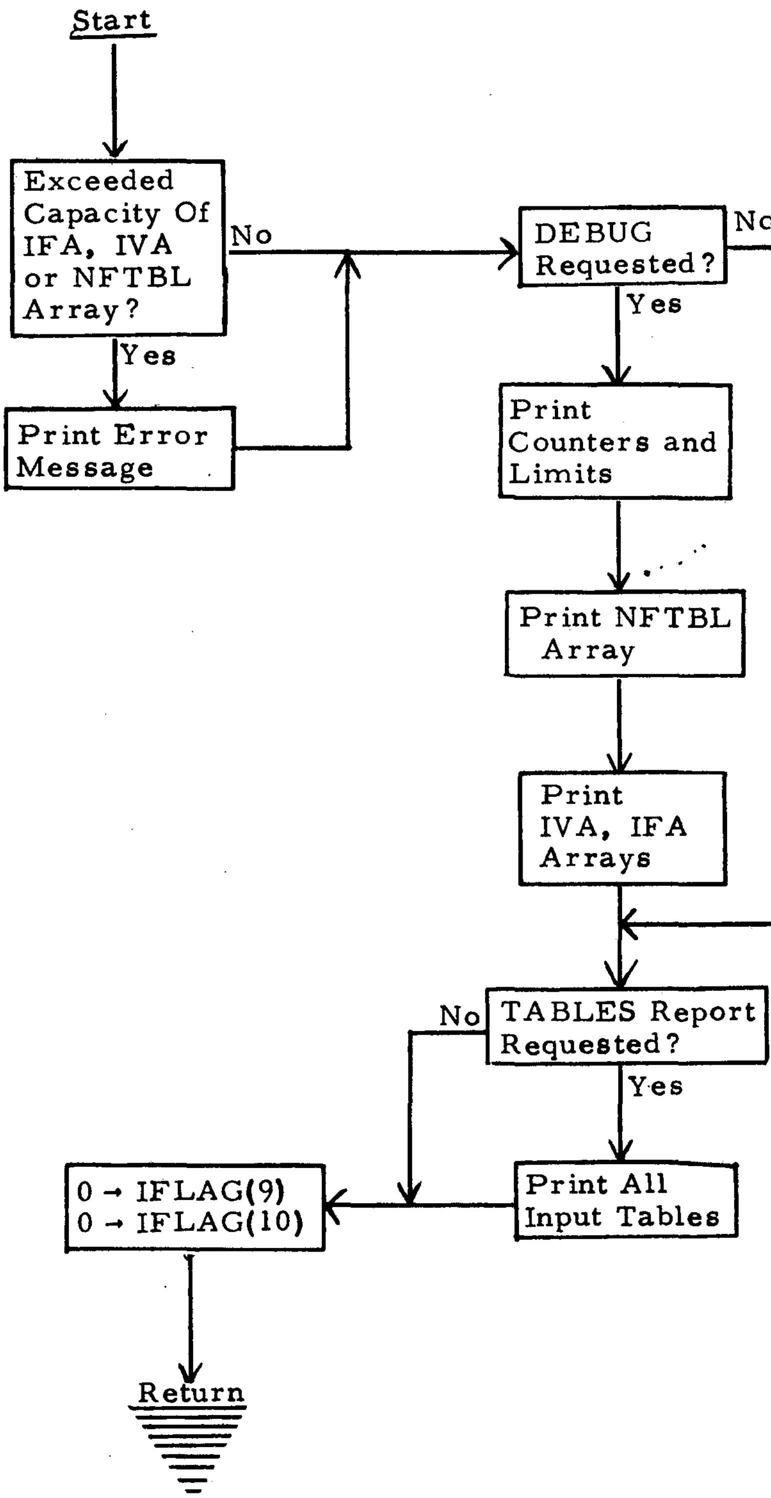
2. NFTBL array.

3. IVA and IFA arrays (vehicle and facility acquisitions)

If IFLAG(7) ≠ 0 (if the TABLES report was requested), all input tables are printed.

Variables IFLAG(9) and IFLAG(10), which trigger certain intermediate printout when nonzero, are zeroed out.

Error messages are printed if the program exceeds the capacity of the IFA, IVA, or NFTBL array.


Start
Exceeded Capacity Of IFA, IVA or NFTBL Array?
No
Yes
Print Error Message
DEBUG Requested?
No
Yes
Print Counters and Limits
Print NFTBL Array
Print IVA, IFA Arrays
TABLES Report Requested?
No
Yes
Print All Input Tables
0 → IFLAG(9)
0 → IFLAG(10)
Return

SECTION 38
SUBROUTINE TRAFIC

TRAFIC serves either or both of two purposes:

1. To print the traffic report for each vehicle giving
   a. The number of flights assigned to each physical vehicle of a given type in each year,
   b. The number of vehicles physically available and also acquired in each year, with running totals,
   c. The number of additional vehicles, above those input, which must be purchased to satisfy cargo shipping schedules.

   This is an optional report obtained by submitting a TRAFFIC card to the input report table.

2. To calculate during leg processing how many (if any) vehicles are needed in addition to those input to satisfy shipping requirements on legs just processed. These extra vehicles are then created and shipped to the lower terminus of the legs they will serve. This process, which is not automatic, is activated by input of a CALVEH card to the input report table.

If neither the CALVEH card nor TRAFFIC card is input, this routine is not called. For the TRAFFIC card, it is called by REPORT; for the CALVEH option, by LEGPRO.

The number of vehicles of a given type needed over some span of years is determined by several variables:

Number of flights in each year	Calculated by program
Number of expended vehicles	
Lifetime of vehicle (number of flights)	Input
Lifetime of vehicle (number of years)	
Maximum number of flights/year	

Since no vehicle is 100% reliable, DORCA is required to maintain the fleet with at least 10% spares at all times, that is, the number of vehicles actually required at any time plus 10%, with fractional parts rounded up to the next highest integer.

A question which had to be resolved was how to allocate flights among available vehicles - whether to use them up quickly by assigning as many flights as possible or to use more vehicles for a longer period of time by assigning fewer flights to each vehicle.

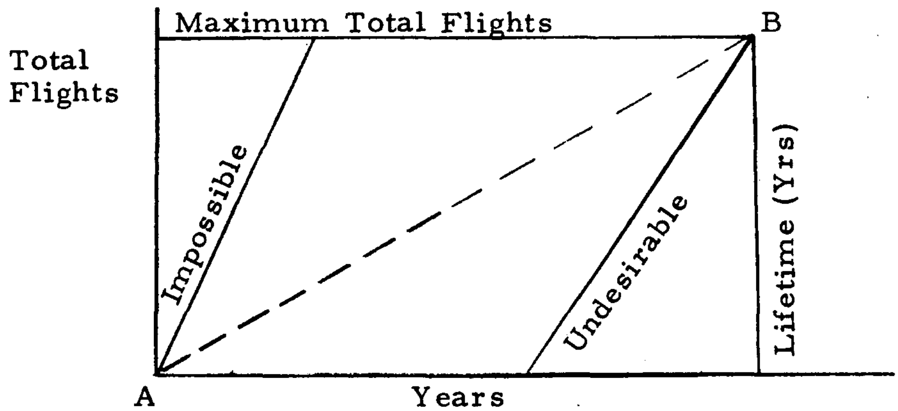


Figure 1

Figure 1 represents a possible graph of a vehicle's activity. One wishes to reach the top line in order to fully utilize each vehicle (usually not possible for all vehicles). The area marked "Impossible" cannot be entered because of the limitation on flights per year. The area marked "Undesirable" should be avoided because, once in it, a vehicle can never attain its maximum total flights due to the annual rate restriction. The ideal path is a straight line from A to B.

In general, TRAFFIC distributes the required flights evenly among the available fleet, including spares. This allocation is modified as necessary to stay within the specified lifetimes of vehicles in both flights and years. Furthermore, if a vehicle is due to expire soon or to be expended, its workload should be increased so that it is used as much as possible before the expended flight or date of expiration. Expended flights are considered to be the last flight assigned to a vehicle in a given year.

This routine distinguishes between a "vehicle" (which is a piece of hardware, a physical unit) and a "type of vehicle" or "vehicle index" (which is a general entry in the vehicle table, a class name for all the physical units). The labor of TRAFIC is done at three levels: (1) generating the tables containing the requirements for all types of vehicles, (2) fulfilling the requirements for a particular type, and (3) assigning flights to each vehicle within its physical limitations.

MTT and FLTAC are the overall tables maintained for all vehicle types. MTT, the Master Traffic Table, is generated from information in the NFTBL array at the beginning of the routine. MTT (I, 1) contains the total number of flights in all years for vehicle type I, where I is the index in the vehicle table. MTT(I, J) contains data for vehicle type I in year J-1 ($2 \leq J \leq 31$) as follows:

Bits 0-17 : number of expended vehicles required
Bits 18-35 : total number of flights

MTT is formed from entries in array NFTBL between the limits NTBL1 and NTBL2. If the CALVEH option has been invoked, TRAFIC is called from LEGPRO to process vehicles for certain groups of legs, at which time NTBL1 and NTBL2 are set in LEGPRO to bound a small subset to NFTBL. When LEGPRO is finished, NTBL1 is reset to 1 and NTBL2 to the last entry, so that the entire NFTBL array is processed. NVMAX is set to the last vehicle type actually entered into the MTT matrix, since not all vehicle types may be active on a given call to TRAFIC. NVMAX $\leq$ NVHMAX = 30.

The FLTAC matrix is created from the vehicle acquisition table IVA which was initiated in RDMISS and possibly undated in previous calls to TRAFIC. FLTAC(I, J) is the number of vehicles of type I acquired by the program and by input in year J-1 ($2 \leq J \leq 31$). FLTAC(I, 1) is the total for all years. MTT and FLTAC are indexed the same and used during subsequent processing.

Each vehicle type is scheduled independently of all others. IVA is the index of the vehicle type currently being scheduled. TOTFLT(I) is the total number of vehicles of this type in the fleet during year I ($1 \leq I \leq 30$). NFR(J) is the number of flights remaining in the lifetime of vehicle number J ($1 \leq J \leq$ MAXVEH). MAXVEH is the maximum number of physical vehicles of any given type which the routine is capable of handling. Current dimensions restrict MAXVEH to available blank common/32. DVTT, the Detailed Vehicle Traffic Table, is the working matrix which will contain the schedules of all vehicles of type IV.

DVTT(I, 1) = total number of flights by vehicle I in all years ($1 \leq I \leq$ MAXVEH)

DVTT(I, J) = number of flights by vehicle #I in year J-1 ($2 \leq J \leq 31$)

$$\text{DVTT}(I, 32) = \begin{cases} 0 & \text{if vehicle is not yet active} \\ 1 & \text{if vehicle is active} \\ 2 & \text{if vehicle has been retired due to age or total flights} \\ 3 & \text{if vehicle is active but will be expended at end of this year} \end{cases}$$

Retired or expended vehicles are deleted from DVTT to conserve space. Printing of the vehicle's activity occurs before the physical deletion.

Initially, TOTFLT, NFR and DVTT are zeroed out for each vehicle type. Each vehicle of type IV in the FLTAC matrix is accumulated into TOTFLT in the year in which it is acquired and, for the time being, assumed to last through the end of the 30-year period. Later on, TOTFLT is updated as the program adds vehicles and deletes those that have expired or been expended.

When a new vehicle is activated, the corresponding entry in the NFR array is set to the maximum total lifetime flights. Three pieces of information are extracted from the input data for vehicle type IV:

MAXPY - maximum number of flights/year allowed
MTF - maximum total flights in lifetime of a vehicle
MNYRS - lifetime of vehicle in years

The program schedules all flights of vehicle IV for one year before going on to the next year.

IYR is the year for which TRAFIC is currently scheduling flights ($1 \leq IYR \leq NYRS \equiv 30$). NFLTS, the number of flights of vehicle type IV for this year, is extracted from MTT. NRV is the number of required (or remaining) vehicles to perform these flights, obtained by dividing NFLTS by MAXPY, rounding upwards, and adding 10% spares. If the number of vehicles already available in the fleet at this time (NVIF) exceeds NRV, then NRV is taken as NVIF. If the number of vehicles to be expended (NE1, obtained from MTT) exceeds NRV, then NRV is upped to that quantity.

Processing begins with a search of DVTT for the first vehicle which is not retired. If NRV exceeds the number of active vehicles in DVTT, or if the number of remaining flights for the active vehicles is less than NFLTS, then it will be necessary to purchase and activate more vehicles. For each vehicle acquired, the following steps are performed:

1) A message is printed (but not in the CALVEH mode)

2) In the CALVEH mode only (when TRAFIC is called by LEGPRO):

   a) The vehicle is added to the phase I cargo table for shipment on lower legs. Variables VEH1 and VEH2 are packed words partially prepared by LEGPRO, transmitted through COMMON, and fully packed and stored by TRAFIC in the standard cargo table format.
   b) A new entry is added to the IVA table.

3) A vehicle is added to FLTAC to this year and to NVIF. A vehicle is added to TOTFLT for this and all subsequent years.

4) A new vehicle is added to DVTT with a status flag of 1 (3 if an expended vehicle is required) and the corresponding entry in the NFR array is taken equal to MTF.

This section describes the method of determining the number of Flights To Be Assigned (NFTBA) to a single vehicle, number IVTL, in year IYR. A first guess for NFTBA is obtained by dividing the current value of flights to be assigned (NFLTS) by the number of remaining vehicles (NRV), rounded upwards. (NFLTS and NRV are both reduced as each group of flights is assigned.)

But various adjustments may be necessary. TRAFIC determines the year in which the vehicle first became active (NYA), the number of years it has been active including the current year (NYACT), and the number of years left in its lifetime after this year (NYLEFT). If NYLEFT < 0, the vehicle expired last year, NFTBA is set to zero, and vehicle IVTL is retired immediately. If NYLEFT = 0, the vehicle will expire at the end of the current year, and NFTBA is set to use all its remaining flights, but not more than MAXPY or NFLTS. Likewise, vehicles to be expended this year are given as many flights as possible (the expended flight is considered to be at the year's end).

If NYLEFT > 0, the program computes how many flight of the maximum permissible total will be unused if the vehicle is flown at the maximum annual rate throughout the rest of its allotted life after this year; this unused quantity is called NEXTRA. (This calculation determines if the vehicle schedule is about to enter the area marked "undesirable" in Figure 1.) If NEXTRA > 0, NFTBA is increased by that amount, but not to exceed the annual limit MAXPY nor the number of remaining flights in its lifetime NFR (IVTL).

At this point NFTBA is determined for vehicle number IVTL. In its bookkeeping, TRAFIC performs the following operations:

1. Inserts NFTBA in DVTT in the slot for the current year and adds it to the total for all years.

2. Subtracts NFTBA from NFR (IVTL) and from NFLTS.

3. Retires the vehicle if it has expired (allotted years are up, vehicle is expended, or number of its remaining flights has been reduced to zero). This involves printing the schedule for vehicle IVTL, subtracting 1 from NVIF, reducing TOTFLT by 1 for this and all subsequent years, and deleting this vehicle from DVTT. The variable KOUNT keeps track of how many expired vehicles have been printed. Expended vehicles have an E printed in column one of the output.

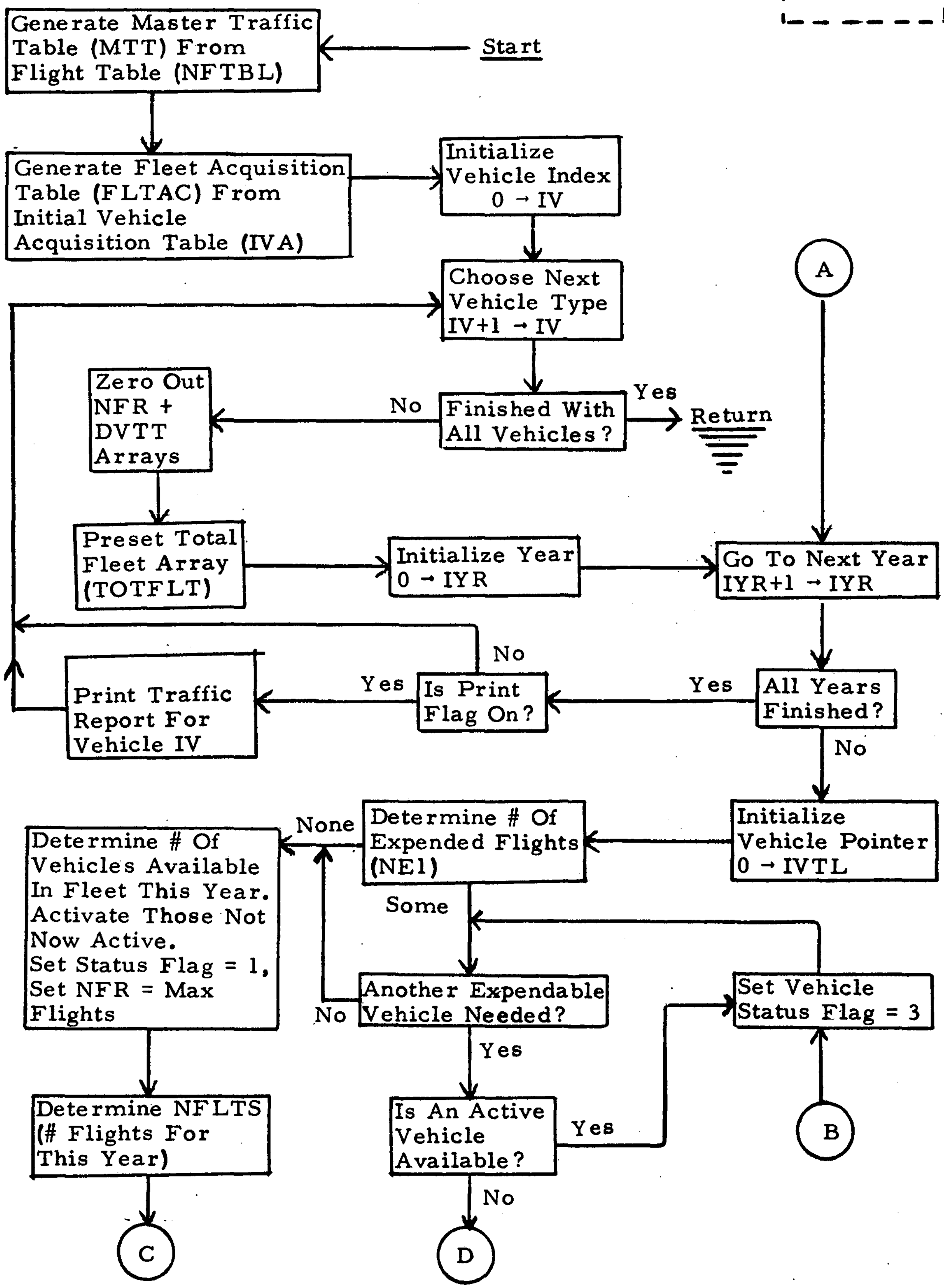
4. Exits for this year if now NFLTS = 0.

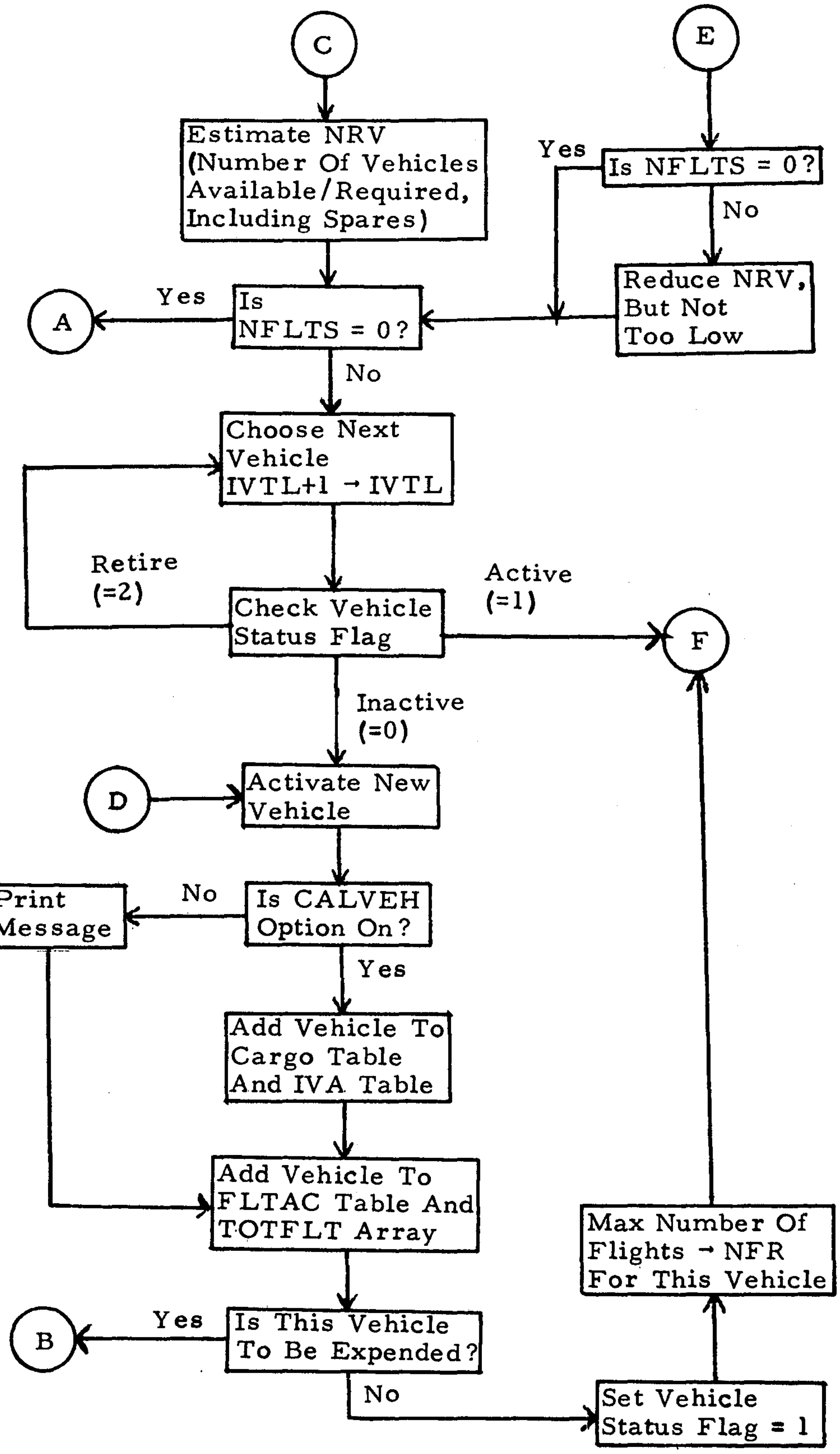
5. Reduces NRV (number of remaining vehicles in fleet) by 1, but to a value not less than 1.

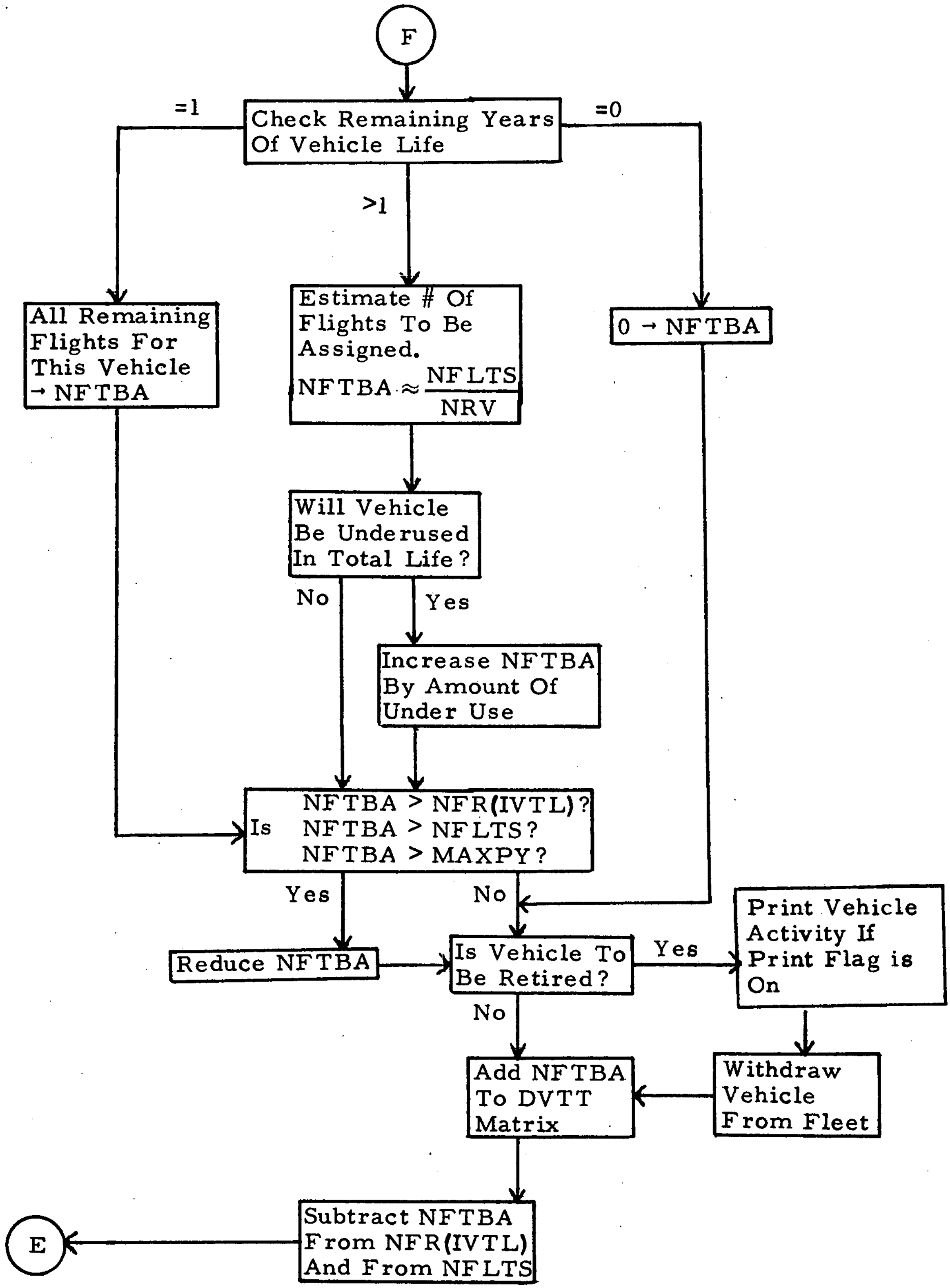
If NFLTS = 0, all flights for this year have been assigned, and TRAFIC goes on to the next year. If NFLTS > 0, TRAFIC repeats this process for vehicle IVTL + 1.

If the traffic report was requested by input, variable IFLAG(4) will contain the value 1. After the complete schedule has been determined for the current vehicle type, TRAFIC prints all remaining (non-expired) vehicles left in DVTT.

TRAFIC


Start
Generate Master Traffic Table (MTT) From Flight Table (NFTBL)
Generate Fleet Acquisition Table (FLTAC) From Initial Vehicle Acquisition Table (IVA)
Initialize Vehicle Index 0 → IV
Choose Next Vehicle Type IV+1 → IV
Finished With All Vehicles?
Yes
Return
No
Zero Out NFR + DVTT Arrays
Preset Total Fleet Array (TOTFLT)
Initialize Year 0 → IYR
A
Go To Next Year IYR+1 → IYR
All Years Finished?
Yes
Is Print Flag On?
No
Yes
Print Traffic Report For Vehicle IV
No
Initialize Vehicle Pointer 0 → IVTL
Determine # Of Expended Flights (NE1)
None
Some
Determine # Of Vehicles Available In Fleet This Year. Activate Those Not Now Active. Set Status Flag = 1, Set NFR = Max Flights
Another Expendable Vehicle Needed?
No
Yes
Set Vehicle Status Flag = 3
Is An Active Vehicle Available?
Yes
No
B
Determine NFLTS (# Flights For This Year)
C
D


C
E
Estimate NRV (Number Of Vehicles Available/Required, Including Spares)
Yes
Is NFLTS = 0?
No
Reduce NRV, But Not Too Low
Yes
A
Is NFLTS = 0?
No
Choose Next Vehicle IVTL+1 → IVTL
Retire (=2)
Active (=1)
Check Vehicle Status Flag
F
Inactive (=0)
D
Activate New Vehicle
Print Message
No
Is CALVEH Option On?
Yes
Add Vehicle To Cargo Table And IVA Table
Add Vehicle To FLTAC Table And TOTFLT Array
Max Number Of Flights → NFR For This Vehicle
B
Yes
Is This Vehicle To Be Expended?
No
Set Vehicle Status Flag = 1


F
=1
Check Remaining Years Of Vehicle Life
=0
>1
All Remaining Flights For This Vehicle → NFTBA
Estimate # Of Flights To Be Assigned.
NFTBA ≈ NFLTS / NRV
0 → NFTBA
Will Vehicle Be Underused In Total Life?
No
Yes
Increase NFTBA By Amount Of Under Use
Is NFTBA > NFR(IVTL)? NFTBA > NFLTS? NFTBA > MAXPY?
Yes
No
Reduce NFTBA
Is Vehicle To Be Retired?
Yes
Print Vehicle Activity If Print Flag is On
No
Add NFTBA To DVTT Matrix
Withdraw Vehicle From Fleet
Subtract NFTBA From NFR(IVTL) And From NFLTS
E

SECTION 39
SUBROUTINE UNPACK

UNPACK is a utility routine to extract bits from a given portion of a 36-bit word. UNPACK extracts NBITS bits from packed word PW starting at bit number B and stores them in V, right adjusted. Bit positions are numbered 0 through 35 from left to right.

To unpack or extract bits, CALL UNPACK(V, NBITS, PW, B)

where NBITS is the number of bits to be extracted.

PW is the word from which bits are to be extracted.

B is the number of the first bit to be extracted.

V is the variable in which the extracted bits are to be stored, right-adjusted as a positive integer. Previous contents of V are destroyed.

SECTION 40
SUBROUTINE VALUE

VALUE converts numeric data from Hollerith or display coded format to floating point format.

The coded value is assumed to be stored in two successive cells in (A6, A4) format, representing 1 to 10 digits with an optional decimal point. The value may be located anywhere in the 10-character field but must not contain embedded blanks. No exponents or plus or minus signs are permitted. The calling sequence is:

CALL VALUE(A, V, IERR)

where A is a 2-cell array containing the coded value in (A6, A4) format.

V contains the converted value in floating point, upon return.

IERR is an error flag, upon return.

IERR = 0 for no error.

IERR = -1 if the field was completely blank. (A blank field is not taken to represent zero.)

IERR = 1, 2, . . . , 10 to indicate position of an illegal character, multiple decimal points or embedded blank.

The purpose of this routine is to detect input errors in numerical entries without causing a machine abort. Thus, this routine examines the input value while it is still in coded format to determine if there are any illegal aspects that would cause an abort. The steps involved in this examination are:

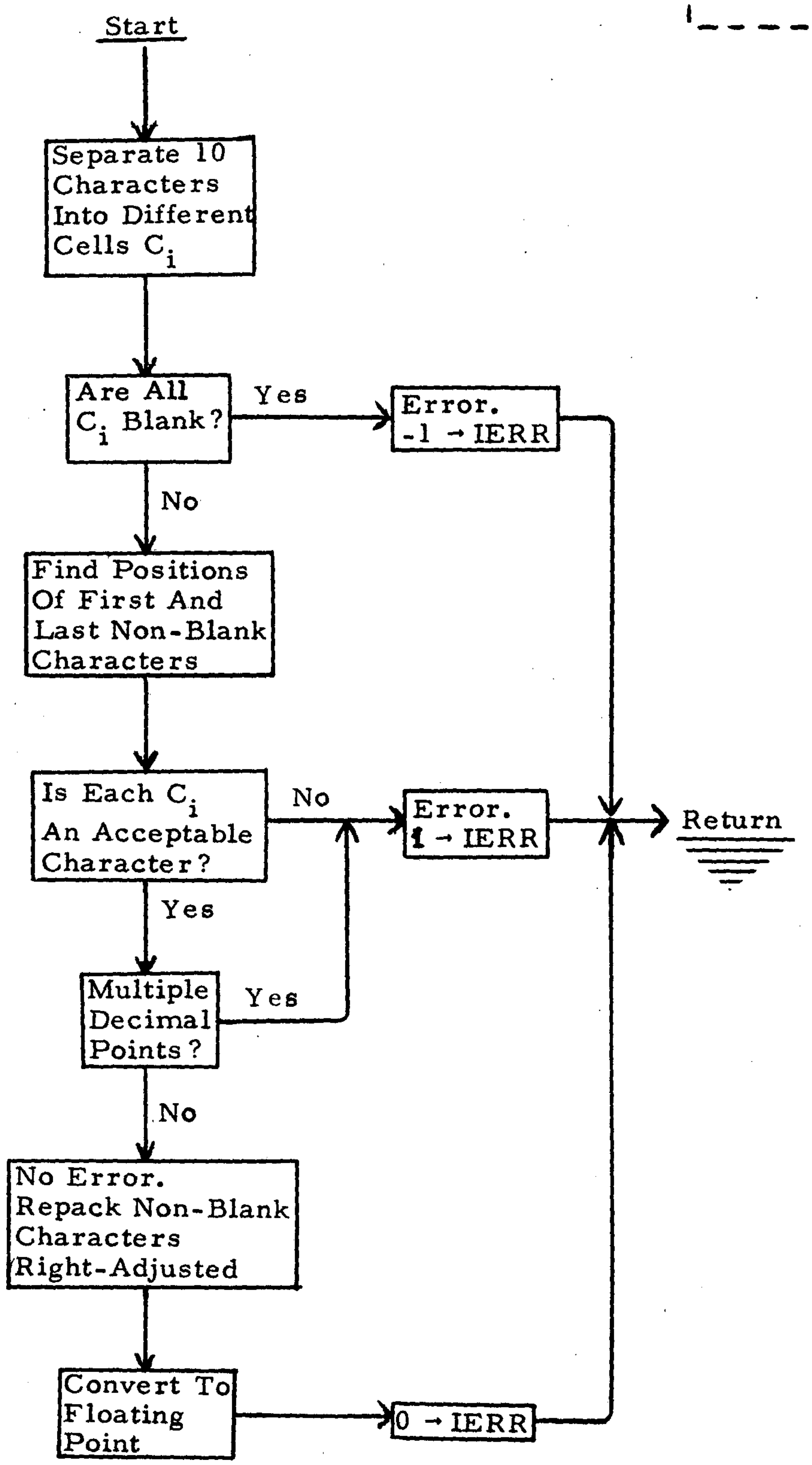
1. Separate the 10 characters into separate machine words.
2. Determine the position of the first and last non-blank characters. If none exist, exit after setting the error flags.
3. Check all characters between the first and last non-blank ones against a list of acceptable characters, which are only the digits 0 through 9

and the decimal point. If there are any illegal characters, embedded blanks or more than one decimal point, set the error flag to the position of the offending character and exit.

4. Repack the individual characters into two words in A6, A4 format, but with the non-blank characters shifted to the extreme right of the field. Convert to floating point format.

The unpacking and repacking of the coded word into 10 separate characters and the ultimate conversion to floating point are accomplished by calling the system routines ENCODE and DECODE, which operate similarly on both the CDC 6000 and Univac 1108 machines, though with slightly different calling sequences. The array C in which the 10 separate characters are stored consists of 19 cells to facilitate right-adjusting the non-blank characters, as is necessary before converting to floating point.

VALUE



Start
Separate 10 Characters Into Different Cells C_i
Are All C_i Blank?
Yes
Error. -1 → IERR
No
Find Positions Of First And Last Non-Blank Characters
Is Each C_i An Acceptable Character?
No
Error. 1 → IERR
Return
Yes
Multiple Decimal Points?
Yes
No
No Error. Repack Non-Blank Characters Right-Adjusted
Convert To Floating Point
0 → IERR

SECTION 41
SUBROUTINE VEHLDF

VEHLDF sums up cargo load factors for each year for a specified vehicle, mission and program.

VEHLDF operates on the phase II cargo table stored in DDB(NBDDB) through DDB(NLDDB). The format of the cargo table and a definition of the term "load factor" is given in Appendix B. VEHLDF is called by VEHRPT and CSTRPT via the following calling sequence:

CALL VEHLDF(VDATA, NB, NL, IPRO, IMIS, IVEH, IFLG)

where VDATA is an array to be filled with load factor sums.

NB indicates the starting location in DDB of cargo items that belong to this program, mission and vehicle (preset by VEHRPT).

IPRO Program number (preset by VEHRPT).

IMIS Mission number (preset by VEHRPT).

IVEH Vehicle number (preset by VEHRPT).

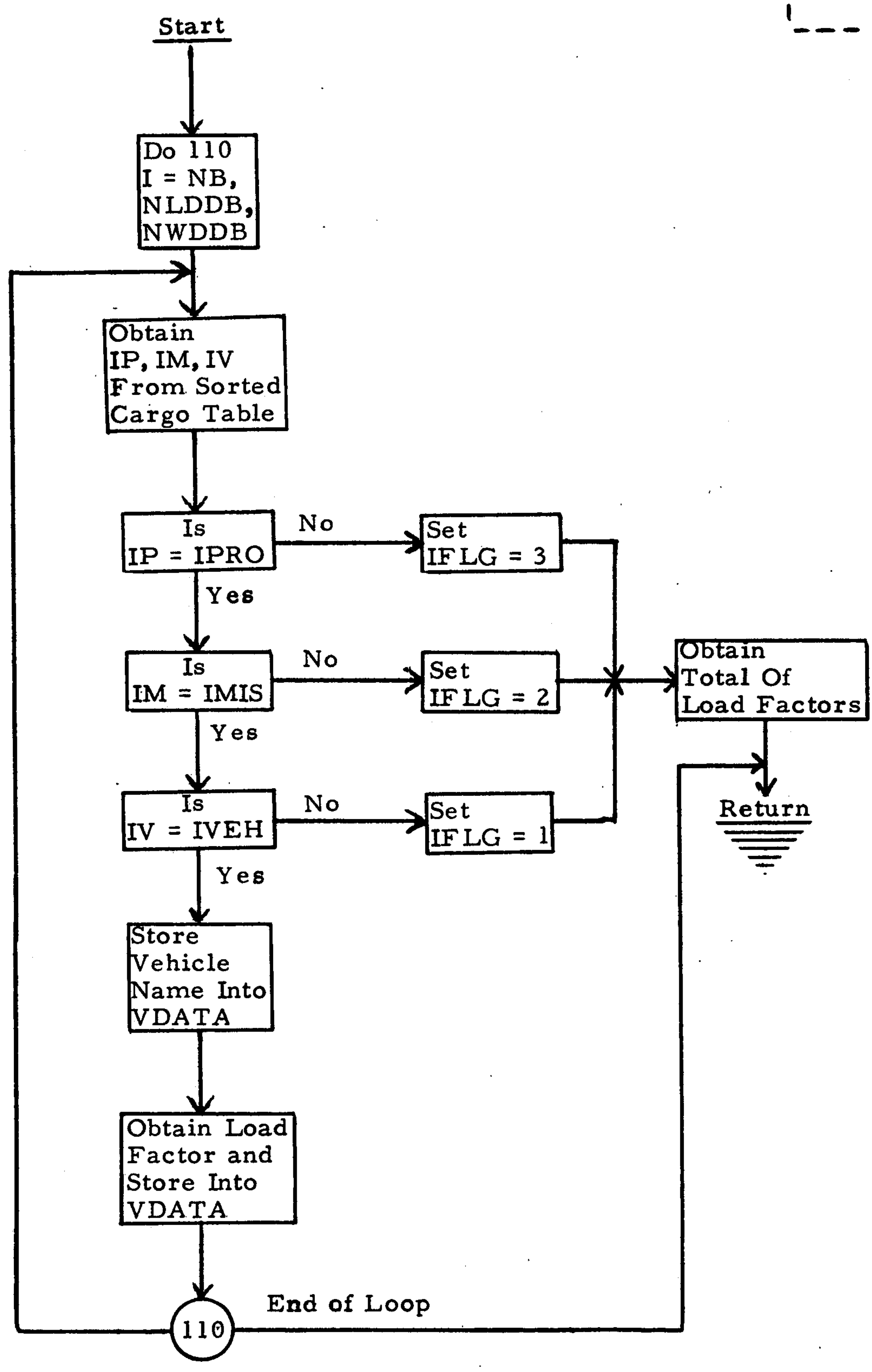
IFLG Is a print control flag to be set by VEHLDF.

The cargo table has been presorted according to program, mission and vehicle in subroutine REPORT. Starting at location NB, the routine looks at the program, mission, and vehicle numbers of each cargo item. If these numbers agree with IPRO, IMIS and IVEH, the routine extracts the load factor and data of the cargo item and accumulates it in array VDATA, whose format is described in subroutine VEHRPT.

As soon as a cargo element is found whose program, mission and/or vehicle are not the required ones, the search is terminated, since the cargo table had been presorted, grouping together all cargo items of the same program, mission and vehicle. Variable IFLG is set to 3, 2, or 1 depending on whether the parameter which changed was the program, mission or vehicle (in that order), and will be used to control printout in VEHRPT. The load factors are now summed for all years and inserted into VDATA(3). Variable NL is now set to the location of the last cargo item of this program, mission, and year, information which is used by VEHRPT to update NB for the next call to VEHLDF. Note that NB and NL each point to the middle word of the 3-word group for a cargo item.

F

VEHLDF


Start
Do 110
I = NB,
NLDDB,
NWDDB
Obtain
IP, IM, IV
From Sorted
Cargo Table
Is
IP = IPRO
No
Set
IFLG = 3
Yes
Is
IM = IMIS
No
Set
IFLG = 2
Obtain
Total Of
Load Factors
Yes
Is
IV = IVEH
No
Set
IFLG = 1
Return
Yes
Store
Vehicle
Name Into
VDATA
Obtain Load
Factor and
Store Into
VDATA
End of Loop
110

SECTION 42
SUBROUTINE VEHRPT

VEHRPT prints the Vehicle Utilization and/or Vehicle Summary reports, which are optional reports requested by input.

The vehicle utilization reports lists the sums of all cargo load factors in each year, broken down by program, mission and vehicle. The vehicle summary is a shorter summation of the same data, a breakdown by vehicle and year only. A definition of the term "load factor" is given in Appendix B.

VEHRPT calls subroutine VEHLDF to sum up the load factors for each year for the current program number IPRO, mission number IMIS, and vehicle number IVEH. These numbers and factors are extracted from the phase II cargo table, whose format is described in Appendix B. Since the cargo table has been sorted by program, mission and vehicle in subroutine REPORT prior to calling VEHRPT, it is not necessary to operate on the entire cargo table at any time but only on a small portion between DDB(NB) and DDB(NL). Variable BN points to the first cargo item and NL to the last cargo item in DDB for this program, mission and vehicle (actually, to the second cell of each 3-word group). Initially, NB is set to the beginning of the cargo table and later updated after NL has been computed by VEHLDF.

The load factors extracted and summed by VEHLDF are stored in array VDATA (equivalent to array LINE) in the following format:

Word	Contents
1-2	Vehicle name (A6, A4 format).
3	Sum of load factors for all years.
4	Sum of load factors for year 1.
5	Sum of load factors for year 2.
6	Sum of load factors for year 3.
⋮	etc.

Data are printed after return from VEHLDF. The data are also accumulated into array VTOT, which has the same format as VDATA but contains the sums by year for all programs and missions. The contents of VTOT are printed under the heading "Vehicle Summary." The total for each year should be identical to the number of flights of that vehicle in that year. Variable NV is a running count on the number of vehicles entered into VTOT so far, since vehicles not assigned to carry cargo in any year are excluded from VTOT.

The last section of the routine prints the vehicle acquisition report.

Array IFLAG is used by DORCA to indicate the various reports requested by input, and IFLAG(5) pertains to VEHRPT:

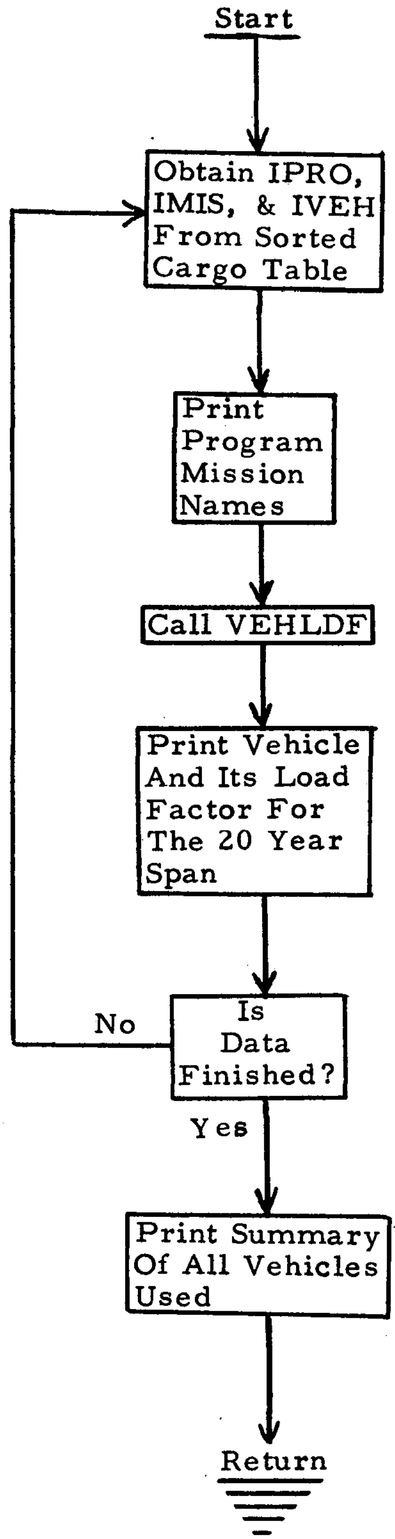
IFLAG(5) =
- 0 No vehicle reports (VEHRPT will not be called).
- 1 Print both vehicle utilization and vehicle summary reports.
- 2 Print vehicle summary report only ("short" report).

The variable IFLG controls program and mission name printouts:

IFLG =
- 1 Print only vehicle name with load factors.
- 2 Print mission name subheading first.
- 3 Print program name heading first, then mission subheading.

IFLG is initially set to 3, then altered in subroutine VEHLDG as it discovers whether the next parameter to change will be the program, mission or vehicle.

VEHRPT



Start
Obtain IPRO, IMIS, & IVEH From Sorted Cargo Table
Print Program Mission Names
Call VEHLDF
Print Vehicle And Its Load Factor For The 20 Year Span
Is Data Finished?
No
Yes
Print Summary Of All Vehicles Used
Return

SECTION 43

SUBROUTINE YEARS

YEARS creates the array NTYRS, which contains the years (last two digits) in which there is nonzero activity (costs incurred, cargo shipped, etc.). YEARS is called from two locations in LEGPRO.

APPENDIX A

MASTER NOMENCLATURE LISTS

This appendix is a master nomenclature list in two parts: 1) all the variables used in DORCA II listed in alphabetical order; and 2) the same variables grouped by the name of the subroutine or labeled common containing the variable.

DORCA NOMENCLATURE LIST

VARIABLE NAME	COMMON OR ROUTINE NAME	DESCRIPTION
A	FACNUM	ARRAY CONTAINING FACILITY NAME (2 WDS), TOTAL ACQUIRED FOR ALL YEARS (1 WD), AND NUMBER ACQUIRED IN EACH YEAR 1970-1999 (30 WDS)
A	FACRPT	AN ARRAY OF THE TOTAL NO. FACILIES PER YEAR EQUIVALENT TO LINE
A	LEGPRO	WEIGHT LOAD FACTOR (FLTG PT)
A	RDMISS	SUM OF UP WEIGHTS OF COMPONENTS OF COUPLED ITEM
A	SORT	ANY MATRIX OF DATA TO BE SORTED. A(MM,NN)
A	VALUE	2-CELL ARRAY CONTAINING CODED VALUE IN (A6,A4) FORMAT
ACP	SPRINT	SUBTOTAL OF FLIGHTS AND VOLUME/LOAD FACTORS FOR THIS VEHICLE/YEAR, ALL LEGS
ACT	SPRINT	GRAND TOTAL OF FLIGHTS AND VOLUME/LOAD FACTORS FOR THIS YEAR
ARRAY	PERLNK	ARRAY OF 8 WORDS FOR EACH OF 1 - 6 VEHICLE STAGES: 1 - WSD, 2 - WNUP, 3 - WPMAX, 4 - WINT, 5 - WPRO, 6 - WNIE, 7 - WACP, 8 - ISP ARRAY OF 8 WORDS FOR EACH OF 1 - 6 VEHICLE STAGES: 1 - WSD, 2 - WNUP, 3 - WPMAX, 4 - WINT, 5 - WPBO, 6 - WNIE, 7 - WACP, 8 - ISP
ARRAY	PROLNK	SUM OF ISP NUMBERS FOR ALL STAGES (TOTAL SPECIFIC IMPULSE FOR VEHICLE)
ARRAY	PROPCL	ARRAY OF 8 WORDS FOR EACH OF 1 - 6 VEHICLE STAGES: WORD 1 - WSD (DRY STRUCTURE WEIGHT) 2 - WNUP (NON-USABLE PROPELLANT WEIGHT) 3 - WPMAX (MAXIMUM PROPELLANT WEIGHT) 4 - WINT (INTERSTAGE WEIGHT) 5 - WPBO (BOIL-OFF WEIGHT) 6 - WNIE (NON-IMPULSIVE PROPELLANT WEIGHT) 7 - WACP (ATTITUDE CONTROL PROPELLANT WEIGHT) 8 - ISP NUMBER (SPECIFIC IMPULSE)
B	PACK	THE BIT NO. TO START PACK/UNPACKING
B	RDCONT	A CONSTANT BLANK (=1H) USED FOR CHECKING FOR SPACES
B	RDCRG	A CONSTANT BLANK (=1H) USED FOR CHECKING FOR SPACES
B	PDMISS	SUM OF DOWN WEIGHTS OF COUPLED COMPONENTS
BCELF	LEGPRO	BULK CARGO LOAD FACTOR
BLANC	RDVEH	EQUAL TO 1H FOR COMPARISON IN FINDING BLANKS
BLANK	RDFAC	EQUAL TO 1H USED TO CHECK FOR BLANKS
BLANK	RDSPD	2-CELL ARRAY CONTAINING FIRST FIELD OF INPUT CARD. EQUIVALENT TO CARD(1-2).
BLANK	RDVEH	EQUIVALENT TO FIRST 10 COLUMNS OF INPUT CARD TO TEST FOR BLANK FIELD.
BLANK	READER	WORD CONTAINING CODED BLANK
BLANK	SPRINT	BLANK WORD
BLF	SPRINT	BULK LOAD FACTOR.
BLKLIM	ASTNER	THIS VARIABLE WHEN APPLIED TO A BULK CONTAINER CAPACITY INDICATES THE MIN. LOAD (LBS.) WHICH MAY BE SHIPPED IN A BULK CONTAINER.
BOX	LEGPRO	CONTAINS CODED WORD -GB BOX- FOR NAME OF NEWLY CREATED CARGO ELEMENTS

BOX2	LEGPRO	CONTAINS CODED WORD -NL BOX- FOR NAME OF NEWLY CREATED CARGO ELEMENTS
C	RDMISS	SUM OF VOLUMES OF COUPLED COMPONENTS
C	VALUE	A 19 CELLED STOAGE OF EACH CHARACTER IN THE CODED VALUE A
CAPAC	RDCONT	WEIGHT CAPACITY FOR CONTAINER EQUAL TO CARD(3)
CARD	/MISC/	16-CELL ARRAY CONTAINING IMAGE OF LAST CARD READ.
CASUC	INPRO	CONTENTS OF COLUMNS 11-20 OF INPUT CARD, USUALLY A TABLE NAME.
CAT	RDCRG	A FLAG WHICH IS SET INDICATING THE CATEGORY OF THE SHIPMENT. (1-MATERIAL, 2-PERSONNEL, 3-FACILITY OR SATELLITE, 4-VEHICLE)
CATEG	RDCRG	CONTAIN CARD IMAGE OF WORD DENOTING ITEM CATEGORY EQUIVALENT TO CARD(8)
CCAP	/ASDAT/	UNUSED CONTAINER CAPACITY (DOES NOT DEPEND ON DIRECTION)
CD	/ASDAT/	CD(I,J) TOTAL AMOUNT OF BULK CARGO STILL UNASSIGNED IN DIRECTION I (I = 1,2) FOR CONTAINER TYPE J(J = 1,....,NCONT).
CD2	/ASDAT/	CD2(I,J) = TOTAL WEIGHT OF ALL BULK CARGO FROM CAPTURE BIN OF CONTAINER TYPE J STILL UNASSIGNED IN DIRECTION I.
CE	DORCA	CORE EXTENSION TO DYNAMIC DATA BLOCK (SEE DDB DEFINITION). ARRAY CE IS DIMENSIONED AND DEFINED IN BLANK COMMON ONLY IN MAIN ROUTINE DORCA. IT ADDS 35000 CELLS ONTO DDB. PURPOSE OF CE IS TO FACILITATE CHANGING THE SIZE OF DDB BY ALTERING ONLY THE DIMENSION OF CE AND THE VALUE OF LOWCOR IN DORCA ONLY.
CE	RDMISS	ARRAY CONTAINING INDICES OF THE COMPONENTS OF EACH COUPLED ITEM
CHAR	READER	ARRAY OF 80 CELLS, EACH CONTAINING ONE CHARACTER FROM CURRENT INPUT CARD.
CL	VALUE	10 CELLED STORAGE OF VALES 0-9 FOR COMPARISON
CLASS	RDCONT	CARGO HANDLING CLASSIFICATION EQUAL TO CARD(7)
CLASS	RDCRG	CONTAINER CLASS (1-CREW CAPSULE, 2-BULK CONTAINER, 3-NONE (DISCRETE ITEM), 4-PROPELLANT TANK)
CNUMB	RDMISS	MULTIPLICITY OF CARGO ELEMENT (NUMBER OF UNITS SHIPPED)
COLS	MERGE	NUMBER OF COLUMNS IN MATRIX FILE WHICH CONTAIN DATA TO BE MERGED.
CONT	LEGPRO	PACKED FIRST WORD FOR CONTAINER BEING ADDED TO PHASE I CARGO TABLE
CONTN	RDCRG	CONTAINS THE NAME OF THE CONTAINER IN WHICH A CARGO WILL BE CARRIED IN. EQUIVALENT TO CARD(6)
CONTRE	FIND	IF CONTRE = 0, BULK CONTAINERS ARE RETURNED (EMPTY) IF CONTRE ≠ 0, CONTAINERS ARE EXPENDED
CONT2	LEGPRO	PACKED SECOND WORD FOR CONTAINER BEING ADDED TO PHASE I CARGO TABLE
COST1	CSTRPT	COST ARRAY. FORMAT- COST NAME (2 WDS), TOTAL FOR ALL YRS (1 WD), TOTAL COST FOR EACH YEAR, 1970-1999 (30 WDS).
COST2	CSTRPT	COST ARRAY SIMILAR TO COST1.
COST3	CSTRPT	COST ARRAY SIMILAR TO COST1.
COST4	CSTRPT	COST ARRAY SIMILAR TO COST1.
COST5	CSTRPT	COST ARRAY SIMILAR TO COST1. NOT USED.
COST6	CSTRPT	COST ARRAY SIMILAR TO COST1.

COUNT	SPDAP	NUMBER OF UNITS PURCHASED IN A GIVEN YEAR
CR	/ASDAT/	TABLE OF CONTAINERS REQUISITIONED
CUTOFF	ASINER	IF VCAP < CUTOFF, VEHICLE IS CONSIDERED FILLED. IF CCAP < CUTOFF, CURRENT CONTAINER IS CONSIDERED FULL AND A NEW CONTAINER IS ASSIGNED.
CW	FIND	CONTAINER WEIGHT
C1	ASINER	FIRST PART OF CARGO ITEM NAME
C2	ASINER	SECOND PART OF CARGO ITEM NAME
DATE	RDMISS	DATE HOLDS THE MISSION DATA START ENTRY IN FLOATING POINT.
DATECI	RDMISS	DATECI HOLDS MISSION DATA UPDATE DATE IN FLOATING POINT.
DATEPI	RDMISS	DATEPI HOLDS MISSION DATA PREVIOUS IOC DATA IN FLOATING POINT.
DATESP	RDMISS	DATESP HOLDS THE MISSION DATA SUSTAINING STOP DATE IN FLOATING POINT.
DATEST	RDMISS	DATEST HOLDS THE MISSION DATA SUSTAINING START DATE IN FLOATING POINT.
DB	RDCRG	EQUIVALENCE TO DDB
DDB	/ /	DYNAMIC DATA BLOCK, CONTAINING CARGO TABLES AND MOST INPUT DATA. HAS DUMMY DIMENSION OF 1 CELL IN ALL SUBROUTINES, BUT THE DIMENSION OF ARRAY CE IN BLANK COMMON DEFINED IN MAIN ROUTINE DORCA ADDS 35000 MORE CELLS. THIS FACILITATES CHANGING THE EFFECTIVE SIZE OF DDB BY SIMPLY CHANGING THE VALUE OF VARIABLE LOWCOR AND THE DIMENSION OF CE IN ROUTINE DORCA ONLY.
DESCRP	RDCRG	DESCRIPTOR TO BE USED AS IDENTIFICATION IN THE PRINTOUTS. EQUIVALENT TO CARD(3)
DEVEL	RDFAC	NON-RECURRING DEVELOPMENT COST
DIRECT	/ASDAT/	DIRECTION OF TRAVEL INDICATOR 1, UP: AWAY FROM EARTH 2, DOWN: TOWARDS EARTH
DISCRP	LEGPRO	7-WORD ARRAY OF CODED DESCRIPTION FOR NEWLY CREATED CARGO ELEMENTS
DNMAX	ASINER	MAX. WEIGHT WHICH VEHICLE CAN CARRY DOWNWARD IF IT FLIES UPWARD COMPLETELY EMPTY.
DNMAX	LEGPRO	MAXIMUM PAYLOAD DOWN FOR THIS VEHICLE
DNMAX	SPRINT	MAXIMUM VEHICLE CAPACITY DOWNWARDS
DOWN	SPRINT	DOWN WEIGHT OF CARGO ITEM.
DOWNWT	RDVEH	2-CELL ARRAY CONTAINING INPUT CODED REPRESENTATION OF DNMAX (MAXIMUM PAYLOAD VEHICLE CAN CARRY DOWNWARDS ON THIS LEG)
DV	PROLNK	VELOCITY INCREMENT DELTA V REQUIRED FOR ORBIT CHANGE (LEG TRAVEL)
DVTT	TRAFIC	DETAILED VEHICLE TRAFFIC TABLE. CONTAINS FLIGHTS IN EACH YEAR FOR EACH PHYSICAL VEHICLE PLUS A STATUS FLAG IN COLUMN 32 (0-NOT YET USED, 1-ACTIVE, 2-RETIRED, 3-TO BE EXPENDED)
DVTT	/ASDAT/	FOR A GIVEN TYPE OF VEHICLE, A MATRIX OF FLIGHT ASSIGNMENTS BY VEHICLE NUMBER AND BY YEAR.
DW	LEGPRO	TINY WEIGHT INCREMENT SUCH THAT, WHEN LEGPRO IS SEARCHING THE CAPTURE BIN FOR THE HEAVIEST ITEM, ITEMS OF EQUAL WEIGHT ARE ASSIGNED PRIORITIES AS FOLLOWS- ROUND TRIP SAME VEHICLE FLIGHT (HIGHEST PRIORITY) ROUND TRIP ON DIFFERENT VEHICLES

		DOWN ONLY
		UP ONLY (LOWEST PRIORITY)
DWNWT	RDCRG	DESCRIBES THE DOWN WEIGHT OF THE CARGO ELEMENT IF THE CARGO ELEMENT IS TO BE SHIPPED DOWN THE LEG. EQUIVALENT TO CARD(12)
EXP	SPRINT	CONTAINS WORD -EXP- IF VEHICLE IS EXPENDED, OTHERWISE BLANK
EXP	/ASDAT/	CONTAINS CODED YES OR NO WORD IDICATING WHETHER CURRENT FLIGHT IS SCHEDULED TO BE EXPENDABLE
EXPMAX	ASINER	MAX. WEIGHT WHICH VEHICLE CAN CARRY IN EXPENDED MODE
EXPMAX	LEGPRO	MAXIMUM PAYLOAD UP FOR THIS VEHICLE IF EXPENDED
EXPMAX	SPRINT	MAXIMUM CAPACITY UPWARDS IF VEHICLE IS EXPENDED
EXPND	RDCRG	CONTAINS CARGO VOLUME IN (A6,A4) FORMAT EQUIVALENT TO CARD(14)
EXPND	SPRINT	CONTAINS WORD -EXP-
EXPWT	RDVEH	2-CELL ARRAY CONTAINING INPUT CODED REPRESENTATION OF EXPMAX (MAXIMUM PAYLOAD VEHICLE CAN CARRY UPWARDS ON THIS LEG IF IT IS EXPENDED)
FACT	RDSPD	ARRAY OF 1-5 COST SPREADING FACTORS ON INPUT CARD IN HOLLERITH FORMAT
FACT	VEHLDF	NAME OF LOAD FACTOR
FACTOR	RDSPD	ARRAY OF 1-7 COST SPREADING FACTORS ON INPUT CARD IN HOLLERITH FORMAT
FACTOR	/ASDAT/	FACTOR(I) CONTAINS EQUIVALENT UP-WEIGHT FACTOR FOR A VEHICLE UP FACTOR(1) = 1.0, DOWN FACTOR(2) = UPMAX/DNMAX
FDV	PERLNK	SAFETY FACTORY FOR ENGINE PERFORMANCE DATA
FDV	PROLNK	SAFETY FACTORY FOR ENGINE PERFORMANCE DATA
FILE	MERGE	MATRIX CONTAINING UP TO 10 LOGICAL RECORDS OF DATA TO BE MERGED.
FLAG	FIND	IF NONZERO, INDICATES CONTAINER ASSIGNED AND SHOULD BE ENTERED INTO THE CR TABLE AND ACCOUNTED IN RUNNING TOTALS
FLAGS	LEGPRO	SIX ONE-BIT FLAGS PACKED INTO BOTH LEVEL I AND LEVEL II CARGO TABLES
FLOW	MERGE	INTEGER USED FOR FLOW CONTROL IN ASSIGNED GO TO.
FLTA	/ASDAT/	TABLE OF CARGO FLIGHT ASSIGNMENTS SCHEDULED FOR THIS LEG/VEHICLE/YEAR
FLTAC	CSTRPT	A MATRIX OF VEHICLE ACQUISITIONS BY VEHICLE TYPE AND BY YEAR
FLTAC	TRAFIC	MATRIX OF VEHICLES ACQUIRED IN EACH YEAR
FLTAC	TRAFIC	A MATRIX OF VEHICLE ACQUISIONS BY VEHICLE TYPE AND BY YEAR....
FLTNO	LEGPRO	CURRENT FLIGHT NO.
FOUR	RDCONT	CONSTANT OF VALUE 4.
FYEAR	/YEAR/	FIRST YEAR THAT ANY CARGO IS SHIPPED, NORMALIZED TO RLDATE. E.G.- IF FYEAR = 1, THE JULIAN DATE IS RLDATE+1.
G	PROPCL	GRAVITY CONSTANT (FT/SEC/SEC)
GBVOL	LEGPRO	PAYLOAD VOLUMES UP AND DOWN (2 WDS)
GBWT	LEGPRO	PAYLOAD WEIGHTS (PRIMARY ITEMS ONLY) UP AND DOWN (2 WDS)
GBWTV	LEGPRO	2-WORD LIST OF VEHICLE WEIGHTS UP AND DOWN (GROUND BASE MODE)
I	ASINER	INDEX VARIABLE
I	CNTRPT	CARGO ELEMENT INDEX OF CURRENT CONTAINER
I	CSTRPT	INDEX VARIABLE

I	DORCA	INDEX VARIABLE
I	DPAGER	INDEX VARIABLE
~~I~~	~~FACNUM~~	~~INDEX VARIABLE~~
I	FACRPT	INDEX VARIABLE
I	FIND	INDEX VARIABLE
~~I~~	~~FINDSP~~	~~INDEX VARIABLE~~
I	LEGPRO	INDEX VARIABLE
I	MERGE	INDEX VARIABLE.
~~I~~	~~RDCONT~~	~~SCRATCH VARIABLE~~
I	RDCRG	USED AS AN INDEX
I	RDFAC	INDEX VARIABLE
~~I~~	~~RDLEG~~	~~INDEX VARIABLE~~
I	RDMISS	INDEX VARIABLE
I	RDRPT	INDEX VARIABLE
~~I~~	~~RDSPD~~	~~INDEX~~
I	RDVEH	INDEX VARIABLE
I	SORT	INDEX VARIABLE
~~I~~	~~SPDAP~~	~~INDEX VARIABLE~~
I	SPRINT	INDEX VARIABLE
I	VALUE	INDEX VARIABLE
~~I~~	~~VEHLDF~~	~~INDEX~~
I	VEHRPT	INDEX PARAMETER
IA	SPRINT	COUNTER TO DETERMINE POSITION IN CARGO TABLE
~~IB~~	~~SPRINT~~	~~LOOP VARIABLE~~
IBOX	LEGPRO	USED TO CREATE UNIQUE SYMBOL FOR NEW COUPLED CARGO BOX NAME
IBVEH	LEGPRO	BEGINNING OF DATA CURRENTLY BEING CONSIDERED
~~IBVEH~~	~~TABLES~~	~~POINTER TO BEGINNING OF DATA FOR THIS VEHICLE IN DDB~~
ICARGO	RDMISS	ARRAY OF 3 PACKED WORDS TO BE ENTERED INTO LEVEL I CARGO TABLE
ICAT	LEGPRO	CATEGORY OF CARGO ELEMENT
~~ICB~~	~~LEGPRO~~	~~ICB=1 IF THERE IS A CAPTURE BIN FOR THIS LEG/YR, =0 IF NOT~~
ICC	LEGPRO	BITS 0-1 OF WORD 3 OF LEVEL I CARGO ITEM (00-REGULAR NON-COUPLED ITEM, 01-IMPOSSIBLE, 10-WHOLE COUPLED COMPOSITE, 11-COMPONENT OF COUPLED ITEM)
~~ICE~~	~~CNTRPT~~	~~CARGO ELEMENT INDEX OF ITEM IN CARGO TABLE~~
ICE	CSTRPT	LOC. OF CARGO ELEMENT IN DDB TABLE
ICE	FACNUM	LOCATION OF DATA IN DDB FOR CURRENT CARGO ELEMENT
~~ICE~~	~~LEGPRO~~	~~INDEX OF ELEMENT IN CARGO ELEMENT TABLE~~
ICE	RDMISS	NUMBER OF COMPONENTS IN CURRENT COUPLED CARGO ITEM
ICEN	RDMISS	POINTER TO LAST INDEX IN ARRAY CE TO DEFINE COMPONENTS OF COUPLED ITEM
~~ICE0~~	~~RDMISS~~	~~ICE0+1 POINTS TO FIRST INDEX IN ARRAY CE LISTING COUPLED COMPONENTS~~
ICF	ASINER	BEGINNING OF CAPTURE BIN IN CARGO TABLE
ICF	LEGPRO	BEGINNING OF CAPTURE BIN IN LEVEL I CARGO TABLE
~~ICF~~	~~RDMISS~~	~~ICF=1 INDICATES ROUTINE IS CURRENTLY PROCESSING A COUPLED ITEM. ICF=0 OTHERWISE.~~
ICGN	LEGPRO	COMPOSITE GROUP NUMBER

ICI	LEGPRO	LOOP COUNTER FOR LOOP THROUGH A PRIORI CARGO LIST
ICL	ASINER	END OF CAPTURE BIN IN CARGO TABLE
ICL	LEGPRO	END OF CAPTURE BIN IN CARGO TABLE
ICLS	LEGPRO	CONTAINER CLASS OF CARGO ITEM
ICNT	CNTRPT	CONTAINER INDEX OF ITEM IN CARGO TABLE
ICNT	/MISC/	NUMBER OF LEVEL II CARGO TABLE ITEMS IN CORE (NOT YET ON TAPE) AT THE MOMENT
ICOL	MERGE	POINTER TO COLUMN OF MATRIX FILE CURRENTLY BEING PROCESSED.
ICPDDB	RDVEH	POINTER TO CURRENT WORD BEING FILLED IN DDB ARRAY
ICPL	LEGPRO	FLAG SHOWING WHETHER LEGPRO HAS (ICPL=1) OR HAS NOT (ICPL) FOUND THE COMPONENTS OF A COUPLED ITEM (THOSE SECONDARIES WITH SAME C.G.N. AS THE PRIMARY)
ICT	/ASDAT/	INDICATES KIND OF CONTAINER REQUIRED BY CURRENT BULK CARGO ITEM
IC2	LEGPRO	BEGINNING OF COUPLED ITEMS (SECONDARIES) FOR THIS LEG/YR. NOT TO BE SHIPPED BY ASINER.
IC3	LEGPRO	END OF COUPLED ITEMS (SECONDARIES) FOR THIS LEG/YR
ID	ASINER	DIRECTION OF CURRENT CARGO ITEM
ID	FIND	DIRECTION OF CURRENT CARGO ITEM
IDATE	LEGPRO	AN INTEGER VALUE OF RLDATE
IDATE	RDMISS	IDATE HOLDS THE MISSION DATA UPDATE DATE MINUS THE RELATIVE DATE
IDATE	TRAFIC	REFERENCE DATE (1970). SAME AS RLDATE
IDC	RDMISS	EQUATED TO DATECI
IDIR	CNTRPT	DIRECTION INDICATOR (0 - UP, 1 - DOWN)
IDIR	LEGPRO	DIRECTION (0-UP, 1-DOWN -OR- 1-UP, 2-DOWN)
IDP	RDMISS	EQUATED TO DATEPI
IDV	LEGPRO	VELOCITY INCREMENT NECESSARY FOR VEHICLES ON THIS LEG
IDV	RDVEH	VELOCITY INCREMENT (DELTA V) FOR THIS LEG
IERR	RDCONT	ERROR FLAG SET IN DECODING INPUT NUMERIC VALUE.
IERR	RDCRG	ERROR FLAG
IERR	RDFAC	ERROR FLAG FROM SUBROUTINE -VALUE- WHILE DECODING NUMERIC INPUT
IERR	RDMISS	ERROR FLAG FROM SUBROUTINE -VALUE-
IERR	RDSPD	ERROR FLAG
IERR	RDVEH	ERROR FLAG FROM SUBROUTINE VALUE
IERR	VALUE	ERROR FLAG IF IERR = 0 NO ERROR -1 COMPLETELY BLANK FIELD 1,2,...,10 POSITION OF ILLEGAL CHARACTER, MULTIPLE DECIMAL POINTS, OR EMBEDDED BLANKS.
IE1	TRAFIC	COUNTER ON EXPENDED FLIGHTS ACCOUNTED FOR SO FAR
IF	ASINER	INDEX IN CARGO TABLE OF FIRST CARGO ITEM FOR THIS VEH/LEG/YR
IF	LEGPRO	LOCATION OF FIRST CARGO ITEM IN DDB FOR THIS LEG/VEHICLE/YEAR
IF	RDMISS	FIRST INPUT CARD WORD WHICH CONTAINS DESCRIPTOR
IFA	/IFA/	FACILITY ACQUISITION LIST
IFAC	CSTRPT	ARRAY INDICATING PROGRAM/MISSION WHICH FIRST USED EACH FACILITY.

IFAC	FACNUM	NO. OF FACILITY BEING CONSIDERED
IFAC	VEHRPT	DVTT(199,1) EQUIVALENCE TO IFAC(1)
~~IFAT~~	~~CSTRPT~~	~~INDEX OF THIS FACILITY.~~
IFAWD	CNTRPT	PACKED WORD TO CREATE ENTRY TO IFA TABLE FOR CONTAINER BEING COSTED
IFD	LEGPRO	FLIGHT NO. DOWN OF CONTAINER
~~IFLAG~~	~~/MISC/~~	~~A FLAG USE IN SPECIFYING REPORTS TO BE PRINTED~~ IFLAG(1) = 1 SPRINT IFLAG(6) = 1 COST IFLAG(2) = 1 CONTAINER IFLAG(7) = 1 TABLES IFLAG(3) = 1 FACILITY IFLAG(8) = 1 DEBUG IFLAG(4) = 1 TRAFFIC IFLAG(9) = 1 CALVEH IFLAG(5) = 1 VEHICLE IFLAG(10) = 1 PRTCAL ~~IFLAG(5) = 2 OR IFLAG(6) = 2 FOR SHORT REPORTS.~~
IFLG	CSTRPT	FLAG TO ROUTE LOGIC FLOW (3 - NEW PROGRAM, 2 - SAME PROGRAM BUT NEW MISSION, 1 - SAME PROGRAM AND MISSION BUT NEW VEHICLE)
IFLG	LEGPRO	FLAG USED TO PACK DATA INDICATING DIRECTION OF CARGO ITEM
IFLG	VEHLDF	FLAG INDEX
IFLG	VEHRPT	FLAG INDEX
~~IFLT~~	~~LEGPRO~~	~~EQUAL TO EITHER IFD OR IFU~~
IFLT	SPRINT	FIRST SET TO LEG/VEHICLE/YEAR/FLIGHT NUMBER OF CURRENT CARGO ITEM. LATER RESET TO PURE FLIGHT NUMBER.
IFOUR	RDVEH	CONTAINS FLOATING POINT CONSTANT 4.0 WITH INTEGER NAME
IFU	LEGPRO	FLIGHT NO. UP OF CONTAINER
IFY	LEGPRO	FIRST YEAR FOR VEHICLE PREFERENCE
~~IGAP~~	~~DORCA~~	~~NUMBER OF UNUSED CELLS IN DDB ARRAY~~
IGAP	LEGPRO	NUMBER OF CORE CELLS FREED BY DELETION FROM CARGO TABLE OF ITEMS (REGULAR AND CAPTURE) JUST SCHEDULED BY ASINER
~~II~~	~~CSTRPT~~	~~TEMPORARY VARIABLE~~
II	LEGPRO	POINTER TO HEAVIEST ITEM LEFT IN CAPTURE BIN
II	SORT	FIRST COLUMN TO BE SORTED (ALWAYS TAKEN AS 1)
~~II~~	~~SPRINT~~	~~NUMBER OF WORDS/LOGICAL RECORD ON TAPE CONTAINING CARGO TABLE~~
II	TRAFIC	LOCATION OF SECOND PART OF VEHICLE NAME, FOR PRINTOUT. ACCOMPANIES LVT.
IJ	SPRINT	LOCATION OF DATA FOR THIS CARGO ITEM IN CARGO ELEMENT TABLE.
~~IL~~	~~ASINER~~	~~INDEX IN CARGO TABLE OF LAST CARGO ITEM FOR THIS VEH/LEG/YR~~
IL	CSTRPT	FLAG INDICATING WHETHER ANOTHER ENTRY SHOULD BE MADE IN IFAC TABLE.
IL	LEGPRO	LOCATION OF LAST CARGO ITEM IN DDB FOR THIS LEG/VEHICLE/YEAR
~~IL~~	~~RDMISS~~	~~LAST INPUT CARD WORD CONTAINING DESCRIPTOR~~
IL	SORT	ARRAY OF LOWER BOUNDS OF SUBINTERVALS TO BE SAVED AND SORTED LATER.
ILEG	CNTRPT	LEG NO. OF CURRENT LEG
~~ILEG~~	~~RDMISS~~	~~ILEG HOLDS A POINTER TO CURRENT MISSION LEG IN LEG TABLE.~~
ILL	LEGPRO	SAVED POINTER TO POSITION IN CARGO TABLE
ILVEH	TABLES	POINTER TO END OF DATA FOR THIS VEHICLE IN DDB
~~ILY~~	~~LEGPRO~~	~~LAST YEAR FOR PREFERENCE OF THIS YEAR~~
IM	ASINER	MODE OF ITEM CURRENTLY BEING PROCESSED.

IM	LEGPRO	RUNNING VARIABLE FOR LOOP ON CAPTURE BIN
IM	VEHLDF	MISSION NAME INDEX
IMAX	LEGPRO	RUNNING INDEX THROUGH CAPTURE BIN
IMIN	MERGE	NUMBER OF COLUMN CONTAINING MINIMUM ELEMENT AND INDICATING DATA TO BE ADDED TO MERGE BUFFER MBUFF.
IMIS	CSTRPT	INDEX TO THE MISSION NAME IN THE MISSION TABLE
IMIS	VEHRPT	INDEX TO THE MISSION NAME IN THE MISSION TABLE.
INDEX	MERGE	INDEX NUMBER OF SCRATCH TAPE CONTAINING DATA TO BE MERGED.
INDEX	SPRINT	COMPOSITE VEHICLE/LEG INDEX FOR XLF MATRIX. INDEX = 100*VEHICLE + LEG
INDEX	/ASDAT/	IF NONZERO UPON RETURN FROM SUBROUTINE -FIND-, A CARGO ITEM SATISFYING REQUIRED MODE, TYPE, DIRECTION, CONTAINER TYPE, VOL. AND WT. LIMIT WAS FOUND AND ASSIGNED TO CURRENT FLIGHT. IF 0, NO SUCH ITEM WAS FOUND.
INIT	INPRO	FLAG INDICATING FIRST TIME THROUGH
INTAPE	MERGE	LOGICAL NUMBER OF TAPE/DISK FILE CONTAINING ORIGINAL INPUT MATRIX.
IN1	MERGE	INDEX NUMBER OF FIRST SCRATCH TAPE CONTAINING SOME DATA.
IN2	MERGE	INDEX NUMBER OF SECOND SCRATCH TAPE CONTAINING SOME DATA.
IOC	RDSPD	SEQUENCE NUMBER OF THE SPREADING FACTOR THAT CORRESPONDS TO THE VEHICLE OR FACILITY IOC DATE. EQUIVALENT TO CARD(5)
IP	ASINER	SCRATCH VARIABLE
IP	MERGE	POINTER TO LOCATION OF DATA ELEMENT IN MINIMUM TEST.
IP	READER	INDEX IN TBVEH MATRIX OF VEHICLE JUST NAMED IN PREFERENCE LIST
IP	VEHLDF	PROGRAM NAME INDEX
IPASS	CNTRPT	FLAG TO DETECT IF ROUTINE IS IN 2ND PASS
IPHASE	RDMISS	IPHASE HOLDS INTEGER CODE TO INDICATE PHASE OF MISSION DATA
IPLTP	/MISC/	LOGICAL NUMBER OF PLOT TAPE
IPM	CSTRPT	PACKED WORD GIVING CURRENT PROGRAM AND MISSION
IPN	RDVEH	WHEN EXTRACTED FROM TBVEH TABLE, CONTAINS NUMBER OF CELLS IN DDB USED FOR DATA FOR THIS VEHICLE
IPO	RDVEH	LOCATION OF START OF DATA FOR THIS VEHICLE IN DDB.
IPREF	LEGPRO	RUNNING INDEX THROUGH VEHICLE PREFERENCE LIST
IPREF	READER	IF IPREF=1, ROUTINE IS PROCESSING VEHICLE PREFERENCE LIST
IPRNT	CSTRPT	ONE OF THE FLAGS WHICH CONTROL WHEN COSTS ARE PRINTED
IPRO	CSTRPT	INDEX TO THE PROGRAM NAME IN THE PROGRAM TABLE
IPRO	VEHRPT	INDEX TO THE PROGRAM NAME IN THE PROGRAM TABLE.
IPROP	/MISC/	THE NO. OF THE CARGO ELEMENT THAT CARRIES ALL THE PROPELLANT IPROP = JPROP + ITBCNT
IPT	FINDSP	LOCATION OF START OF DATA FOR THIS SPREAD FUNCTION IN DDB ARRAY.
IP0	TRAFIC	LOCATION OF START OF INPUT DATA FOR THIS VEHICLE IN DDB
IRELDT	/YEAR/	THE YEAR TO WHICH ALL DATES ARE RELATIVE (I.E.-1970). SAME AS RLDATE BUT IN INTEGER FORMAT.
IROUND	LEGPRO	INTEGER 100, USED TO ROUND PROPELLANT OFF-LOADED UPWARDS TO NEAREST 100 LBS
IRPT	READER	NUMBER OF REPORT REQUESTS READ SO FAR.

IRT	ASINER	1-BIT ROUND-TRIP FLAG IN CARGO TABLE IF IRT = 1, ITEM MUST MAKE ROUND TRIP. IF IRT = 0, ITEM TRAVELS IN ONE DIRECTION ONLY
IRTDN	LEGPRO	COMPOSITE FLAG CONTAINING ROUND-TRIP AND DIRECTION FLAGS OF CURRENT CARGO ITEM (1 BIT EACH). 2-ROUND TRIP, 1-DOWN ONLY, 0-UP ONLY
IRTDN	RDMISS	FLAG INDICATING MODE OF DELIVERY (0=DEPLOY, 1=RETRIEVE, 6=SERVICE, -1=DEFAULT)
IRTN	SPRINT	IRTN = 30 MEANS ENTIRE CARGO TABLE HAS BEEN PROCESSED
IRTN	TRAFIC	INDICATES WHETHER NEWLY ACTIVATED VEHICLE WILL BE EXPENDED THIS YEAR (IRTN=2) OR NOT (IRTN=1)
IS	ASINER	SUBSCRIPT OF CURRENT CARGO ITEM IN DDR ARRAY
IS	FIND	SUBSCRIPT OF CURRENT CARGO ITEM IN DDB ARRAY
IS	LEGPRO	LOCATION OF THIS CARGO AT ITS SOURCE IN PHASE I CARGO TABLE
ISAVE	LEGPRO	TEMPORARY STORAGE FOR CONTENTS OF IFLAG(4), WHICH IS ALTERED
ISD	ASINER	ISD=1 IF CARGO ITEM REQUIRES SINGLE DEPLOYMENT, =0 IF MULTIPLE DEPLOYMENT IS A ACCEPTABLE.
ISD	FIND	INDICATES WHETHER CARGO ITEM REQUIRES SINGLE DEPLOYMENT (ISD=1) OR MULTIPLE DEPLOYMENT (ISD=0)
ISKIP	LEGPRO	ISKIP = 0 FOR VEHICLE UPPER STAGE IN UP DIRECTION 1 FOR COUPLED ITEM UP 2 VEHICLE UPPER STAGE GOING DOWN THE LEG 3 COUPLED ITEM GOING DOWN 4-8 FOR LOWER VEHICLE STAGES
ISP	PROPCL	ARRAY GIVING SPECIFIC IMPULSE FOR EACH STAGE
ISP	RDVEH	CONTAINS CODED WORD ≠ISP≠
ISPEF	PROPCL	ISPEF(I) = EFFECTIVE SPECIFIC IMPULSE FOR STAGE I.
ISPS	PERLNK	SUM OF ISP NUMBERS FOR ALL STAGES (TOTAL SPECIFIC IMPULSE FOR VEHICLE)
IT	ASINER	TYPE OF CURRENT ITEM. SEE ITEM
ITAPE	MERGE	LOGICAL TAPE NUMBER OF SCRATCH FILE CONTAINING SOME DATA.
ITBCNT	/MISC/	THE NO. OF THE LAST CARGO ELEMENT IN CARGO ELEMENT TABLE BEFORE THE CONTAINERS ARE ADDED
ITEMP	RDMISS	ITEMP HOLDS AN INTEGER CODE TO INDICATE DISCRETE OR BULK CARGO. PROCESSING.
ITEMP	TRAFIC	TEMPORARY VARIABLE USED IN EXTRACTING DATA FROM CARGO ELEMENT TABLE
ITEMS	/ASDAT/	LOWER INDEX LIMIT ON UNASSIGNED ITEMS REMAINING IN WS/VOL MATRICES. ACTUAL NO. OF ITEMS REMAINING UNASSIGNED IS MAXI+1-ITEMS
ITEST	READER	WORD CONTAINING FIRST CHARACTER ONLY OF CURRENT CARD. IF BLANK AND IF LAST CARD WAS A COMMENT CARD, THEN THIS ONE IS, TOO.
IU	SORT	ARRAY OF UPPER BOUNDS OF SUBINTERVALS TO BE SAVED AND SORTED LATER.
IV	RDCONT	SCRATCH VARIABLE EQUIVALENCE TO V
IV	SPRINT	LOOP VARIABLE
IV	TRAFIC	INDEX OF VEHICLE TYPE CURRENTLY BEING SCHEDULED
IV	VEHLDF	VEHICLE NAME INDEX

IVA	/IVA/	VEHICLE ACQUISITION TABLE
IVAQ	TRAFIC	NUMBER OF VEHICLES OF THIS TYPE ACQUIRED THIS YEAR REMAINING IN UNDETERMINED STATUS
IVAT	TRAFIC	TOTAL COUNT OF VEHICLES OF THIS TYPE ACQUIRED IN ALL YEARS
IVEH	CSTRPT	POINTER TO BEGINNING OF DATA FOR CURRENT VEHICLE IN DDR. ALSO USED AS A VEHICLE NUMBER.
IVEH	LEGPRO	INDEX OF VEHICLE WET STAGE
IVEH	PROLNK	VEHICLE INDEX
IVEH	RDMISS	IVEH = 0 - NO VEHICLE CARD 1 - A VEHICLE IS SPECIFIED FOR EACH LEG 2 - AT LEAST ONE LEG MUST BE CAPTURED
IVEH	TRAFIC	VEHICLE INDEX FOR LOWER LEGS
IVEH	VEHRPT	INDEX TO THE VEHICLE NAME IN THE VEHICLE TABLE.
IVH	LEGPRO	VEHICLE INDEX EXTRACTED FROM LEVEL I CARGO ITEM
IVOL	RDVEH	VOLUME
IVTL	TRAFIC	NUMBER OF VEHICLE CURRENTLY BEING ASSIGNED FLIGHTS. POINTS TO DVTT MATRIX AND NFR ARRAYS
IVTOT	VEHRPT	= VOTOT(1,1)
IWORD	RDMISS	PACKED WORD TO BE ADDED TO IFW OR IVA ARRAY
IWVEH	TABLES	NUMBER OF WORDS OF DATA FOR THIS VEHICLE IN DDR
IX	ASINER	TEMPORARY VARIABLE
IX	FIND	TEMPORARY VARIABLE
IX	RDSPD	TEMPORARY VARIABLE
IX	TRAFIC	CONTAINS NUMBER OF EXPENDED VEHICLES IN LEFT HALF OF WORD (BITS 0-17)
IY	READER	FIRST YEAR (RELATIVE TO RLDATE) FOR PREFERENCE VEHICLE
IYEAR	SPRINT	YEAR (4 DIGITS)
IYR	CNTPPT	CURRENT YEAR OF FLIGHT, NORMALIZED TO RLDATE.
IYR	TRAFIC	YEAR IN WHICH FLIGHTS ARE ASSIGNED, RELATIVE TO IDATE = RLDATE
IYR	VEHLDF	YEAR INDEX
IYRIOC	SPDAP	YEAR OF SPREAD FUNCTION RANGE IN WHICH ITEM IS DELIVERED.
IYRSP	SPDAP	NUMBER OF YEARS SPANNED BY SPREAD FUNCTION
I1	LEGPRO	LOOP VARIABLE
I1	PERLNK	BIT POSITION TO START UNPACKING
I1	RDMISS	I1 IS A CONSTANT VARIABLE EQUAL TO ONE
I1	SPRINT	PROGRAM NUMBER.
I1	TRAFIC	INDEX FOR UNPACKING STAGES WHEN MORE VEHICLES ARE ACQUIRED AND SHIPPED IN CARGO TABLE
I2	LEGPRO	BIT POSITION FOR UNPACKING STAGE NUMBERS
I2	PROLNK	BIT POSITION FOR UNPACKING
I2	RDCONT	SCRATCH VARIABLE
I2	RDCRG	USED AS AN INDEX
I2	RDLEG	INDEX VARIABLE
I2	RDMISS	LOOP LIMIT USED IN SEVERAL SENSES

I2	RDSPD	INDEX
I2	RDVEH	UPPER LIMIT OF DO LOOP EQUAL TO NVEH-1
I2	SPRINT	MISSION NUMBER.
I2	TRAFIC	BIT POSITION FOR UNPACKING STAGES
I3	SPRINT	LEG NUMBER
I4	SPRINT	VEHICLE NUMBER.
I5	SPRINT	RELATIVE DATE.
I6	SPRINT	FLIGHT NUMBER
I7	SPRINT	CARGO ELEMENT INDEX.
I77	/MISC/	OCTAL CONSTANT 777777777777 USED FOR PACKING. SET IN SUBR. RDVEH
I8	SPRINT	DIRECTIONAL FLAG (0 - UP, 1 - DOWN)
J	ASINER	SCRATCH VARIABLE
J	CSTRPT	INDEX VARIABLE
J	MERGE	INDEX VARIABLE
J	RDCONT	SCRATCH VARIABLE
J	RDCRG	INDEX VARIABLE
J	RDFAC	INDEX VARIABLE
J	RDLEG	INDEX VARIABLE
J	RDMISS	INDEX VARIABLE
J	RDVEH	INDEX VARIABLE
J	SORT	INDEX VARIABLE
J	SPDAP	TEMPORARY VARIABLE
J	SPRINT	INDEX VARIABLE
J	VALUE	INDEX VARIABLE
J	VEHRPT	INDEX VARIABLE
JBOX	LEGPRO	USED TO CREATE UNIQUE SYMBOL FOR NAME OF NEW COUPLED CARGO BOX
JCARGO	RCMISS	ARRAY OF 3 PACKED WORDS TO BE ENTERED INTO LEVEL I CARGO TABLE FOR COUPLED CARGO
JCASE	DORCA	COUNT OF CASES DORCA HAS ATTEMPTED TO EXECUTE
JCGN	LEGPRO	2-WORD LIST OF COMPOSITE GROUP NUMBER UP AND DOWN
JCI	LEGPRO	CONTAINS BITS 0 AND 1 TO BE PACKED INTO WORD 3 OF LEVEL II DATA. 2-WORD ARRAY FOR UP AND DOWN TRIPS
JDONE	/ASDAT/	JDONE IS SET TO 1 WHEN ALL CARGO (UNCLUDING CAPTURE BIN) HAS BEEN ASSIGNED
JERR	/MISC/	VARIABLE USED TO COUNT THE NUMBER OF ERRORS THAT OCCUR.
JF	CSTRPT	POINTER TO CURRENT POSITION IN FACILITY ACQUISITION TABLE (IFA).
JF	FACNUM	POINTER TO CURRENT POSITION IN FACILITY ACQUISITION TABLE IFA
JF	FACRPT	INTEGER POINTER TO FACILITY INFORMATION TO BE CONSIDERED CURRENTLY
JFLAG	REPORT	FLAG INDICATING IF VEHICLE AND/OR COST REPORT IS TO BE PRINTED.
JFLAG	/MISC/	JFLAG=1 ON FIRST PASS THRU COST REPORT (NO PRINTING) =0 ON SECOND PASS (PRINTOUT IS GENERATED)
JFLT	LEGPRO	FLIGHT NUMBER TO INSERT INTO LEVEL II CARGO TABLE.
JFLT	SPRINT	LEG/VEHICLE//YEAR/FLIGHT NUMBER OF LAST CARGO ITEM.
JJ	SORT	LAST COLUMN TO BE SORTED (ALWAYS TAKEN AS NN)
JJ	TRAFIC	YEAR IN WHICH EXTRA VEHICLE IS NEEDED (4 DIGITS FOR PRINTOUT)

JL	ASINER	SCRATCH VARIABLE - UPPER BOUND OF DDB CONTAINING DATA FOR CURRENT VEH.
JL	LEGPRO	END OF DATA FOR THIS VEHICLE IN DDB
JL	RDFAC	SCRATCH VARIABLE USE TO CHECK FOR NEWLY OCCURRED ERROR
JL	SPRINT	POINTS TO END OF DATA FOR THIS VEHICLE IN DDB
JLEG	CNTRPT	LEG NO. OF LEG PRIOR TO CURRENT LEG
JN	LEGPRO	NUMBER OF WORDS OF DATA FOR THIS VEHICLE IN DDB
JPROP	/MISC/	NO. OF THE LAST CONTAINER THAT CAN CARRY A PROPELLANT TANK
JRT	ASINER	JRT=1 IF CARGO ITEM MUST MAKE ROUND TRIP ON SAME VEHICLE (SAME FLIGHT NUMBER) =0 IF ROUND TRIP MAY BE ASSIGNED TO DIFFERENT FLIGHTS.
JRT	FIND	IF JRT=1, CARGO MUST MAKE ROUND TRIP ON SAME FLIGHT
JSPD	RDFAC	POINTER TO SPREAD TABLE FOR DEVELOPMENT COST.
JV	PROPCL	IF JV=1, THIS IS THE FIRST TIME THROUGH THE UPWARDS CALCULATIONS
JVEH	LEGPRO	VEHICLE SELECTED FROM PREFERENCE LIST FOR CAPTURE CARGO IF AN EXPENDED FLIGHT IS NECESSARY.
JVEH	PROLNK	INDEX OF VEHICLE STAGE
JVEH	RDMISS	JVEH = 1 FOR REGULAR CARGO, JVEH = 2 FOR A DISCRETE CARGO TO BE CAPTURED WITH AN UP WEIGHT AND A DOWN WEIGHT.
JWORD	LEGPRO	PACKED WORD IN FORMAT OF THIRD WORD OF LEVEL I CARGO ITEM (COUPLED)
JWORD	RDMISS	PACKED WORD USED TO FORM THIRD WORD OF LEVEL I DATA FOR COUPLED ITEM
JX	PROLNK	BEGINNING OF DATA FOR VEHICLE JVEH IN DDB
JX	RDVEH	FLAG SET TO 1 WHEN AT LEAST THE MINIMUM AMOUNT OF DATA REQUIRED HAS BEEN INPUT FOR THIS VEHICLE.
JX	SPRINT	NUMBER OF LINES ENTERED IN XLF ARRAY
JXMAX	SPRINT	MAXIMUM NUMBER OF VEHICLE/LEG COMBINATIONS WHICH MATRIX XLF CAN ACCOMODATE
JY	PROLNK	BEGINNING OF DATA VEHICLE IVEH IN DDB
JY	READER	LAST YEAR (RELATIVE TO RLDATE) FOR PREFERENCE VEHICLE
JYR	TRAFIC	YEAR IN WHICH ACQUIRED VEHICLE IS ADDED TO PHASE I CARGO TABLE
J1	ASINER	SCRATCH VARIABLE - LOWER BOUND OF DDB CONTAINING DATA FOR CURRENT VEH.
J1	LEGPRO	START OF DATA FOR THIS VEHICLE IN DDB
J1	PERLNK	LOCATION IN DDB WHERE ISP DATA FOR THIS VEHICLE STAGE SHOULD START
J1	PROLNK	LOCATION OF START OF ISP DATA FOR VEHICLE JVEH
J1	SPRINT	POINTS TO BEGINNING OF DATA FOR THIS VEHICLE IN DDB
J77	/MISC/	OCTAL CONSTANT 000000000077 USED FOR PACKING. SET IN SUBR. LEGPRO
K	ASINER	INDEX VARIABLE
K	FIND	INDEX VARIABLE
K	RDFAC	INDEX VARIABLE
K	RDMISS	INDEX VARIABLE
K	RDVEH	INDEX VARIABLE
K	SORT	COLUMN POINTER.
K	SFDAP	INDEX VARIABLE
K	SPRINT	LOOP VARIABLE
K	VEHRPT	INDEX VARIABLE
KALL	PERLNK	NUMBER OF TIMES PROPCL WAS CALLED (NUMBER OF ITERATIONS)

KBOX	LEGPRO	KLUDGE FACTOR TO CREATE COUPLED CARGO BOX NAME
KEL	MERGE	COLUMN ELEMENT ON WHICH SORT IS MADE (SAME AS SUBROUTINE ARGUMENT KELT)
~~KEL~~	~~/KEL/~~	~~NUMBER OF THE MATRIX ELEMENT UPON WHICH THE SORT/MERGE IS PERFORMED~~
KFLAG	CSTRPT	SAME AS IFLAG(11). IF KFLAG=1, REPORT WILL BE PRINTED IN 8X11-INCH FORMAT, OTHERWISE IN 8X14-INCH FORMAT.
~~KK~~	~~SPRINT~~	~~NUMBER OF WORDS OF CARGO TABLE TO READ FROM TAPE~~
KOPT	/MISC/	ARRAY CONTAINING CURRENT STATUS OF INPUT OPTIONS WORD 1 - PROPELLANT OFF-LOADING/FULL TANK ~~2 - SPACE/GROUND BASED~~ 3 - SINGLE/MULTIPLE CARGO DEPLOYMENT 4 - MANUAL/AUTOMATIC VEHICLE DELIVERY ~~5 - CONTAINER RETURN/EXPEND~~ 6 - CARGO FOR CAPTURE BIN/ASSIGNED VEHICLE
KOUNT	TRAFIC	COUNT ON NUMBER OF PHYSICAL VEHICLES PRINTED SO FAR. DIFFERS FROM IVTL, WHICH ~~IS REDUCED WHEN VEHICLES ARE EXPENDED OR RETIRED~~
KSPD	RDFAC	POINTER TO SPREAD TABLE FOR PRODUCTION COST.
KVEH	ASINER	NO. OF CURRENT VEHICLE
~~KVEH~~	~~LEGPRO~~	~~VEHICLE SELECTED WHEN ONLY A CAPTURE BIN IS LEFT FOR THIS LEG/YEAR (NO CARGO~~ WITH A SPECIFIED VEHICLE)
K1	SPRINT	STARTING POINT TO PLACE CARGO TABLE DATA IN DDB
~~L~~	~~ASINER~~	~~SCRATCH VARIABLE~~
L	CNTRPT	LOCATION OF CARGO ITEM IN LEVEL 2 CARGO TABLE
L	CSTRPT	INDEX VARIABLE
~~L~~	~~FACNUM~~	~~INDEX VARIABLE = NYR-FYEAR+4~~
L	RDMISS	INDEX VARIABLE
L	RDVEH	INDEX VARIABLE
~~L~~	~~SORT~~	~~COLUMN POINTER.~~
L	SPDAP	TEMPORARY VARIABLE
L	TRAFIC	INDICATES PROPER SLOT FOR YEAR NY1 IN MTT/FLTAC ARRAYS
~~L~~	~~VEHLDF~~	~~INDEX~~
LASTCE	LEGPRO	INDEX OF LAST CARGO ELEMENT INPUT (NOT CREATED BY LEGPRO)
LASTLG	/LEGS/	POINTER TO LAST (SPECIAL) SLOT IN LEG TABLE
~~LDF~~	~~VEHLDF~~	~~NAME OF LOAD FACTOR~~
LEG	ASINER	NO. OF CURRENT LEG
LEGNM	RDVEH	NAME OF LEG ON WHICH VEHICLE CAN BE USED.
~~LEGNO~~	~~LEGPRO~~	~~LEG NUMBER OF NEXT BLOCK OF CARGO ITEMS.~~
LEG1	LEGPRO	LEG NUMBER OF CURRENT BLOCK OF CARGO ITEMS.
LEN	TABLES	NUMBER OF WORDS OF DATA FOR THIS SPREAD FUNCTION IN DDB
~~LF~~	~~LEGPRO~~	~~100,000 TIMES THE LOAD FACTOR~~
LF	SPRINT	EQUIVALENCED TO XLF
LFA	SPRINT	LOAD FACTOR X 100000, IN INTEGER FORMAT.
~~LFB~~	~~LEGPRO~~	~~100,000 TIMES THE BULK LOAD FACTOR~~
LFB	SPRINT	BULK LOAD FACTOR X 100000, IN INTEGER FORMAT.

LFGB	LEGPRO	LOAD FACTOR * 100000 FOR GROUND BASE OPERATIONS
LGYR1	LEGPRO	LEG/YEAR OF A CARGO ITEM
~~LGYR1~~	~~LEGPRO~~	~~LEG/YEAR OF FIRST CARGO ITEM~~
LIFFLT	RDVEH	MAX NO. FLIGHTS BEFORE REPLACEMENT
LIFYRS	RDVEH	MAX YEARS OPERATION BEFORE REPLACEMENT
~~LIM~~	~~TABLES~~	~~NUMBER OF ENTRIES IN VEHICLE OR FACILITY ACQUISITION TABLE, WHICHEVER IS GREATER~~
LIMCAL	FIND	MAXIMUM NUMBER OF CONSECUTIVE UNSUCCESSFUL CALLS TO SUBROUTINE -FIND- BEFORE PROGRAM ABORTS ITSELF. LIMIT IS 100.
~~LIMLEG~~	ASINER	~~LIMIT ON NUMBER OF CARGO ITEMS WHICH MAY BE ASSIGNED TO ANY FLIGHT OF ANY~~ VEHICLE ON THIS LEG.
LIML2	LEGPRO	MAX NUMBER OF LEVEL II ITEMS WHICH CAN FIT IN 510-WORD BUFFER
~~LIMOCC~~	~~/ASDAT/~~	~~LIMOCC(I) IS THE MAXIMUM NUMBER OF CARGO ITEMS WHICH MAY BE ASSIGNED TO ANY~~ FLIGHT UPWARDS (I=1) OR DOWNWARDS (I=2)
LIMVEH	ASINER	LIMIT ON NUMBER OF CARGO ITEMS WHICH MAY BE ASSIGNED TO ANY FLIGHT OF THIS VEHICLE.
LINE	CNTRPT	ARRAY OF ANNUAL COUNTS FOR CURRENT CONTAINER AND LEG
LINE	DPAGER	ARRAY CONTAINING COST DATA FOR EACH YEAR
~~LINE~~	~~TRAFIC~~	~~ARRAY CONTAINING ANNUAL COUNTS OF VEHICLES ACQUIRED TO DATE~~
LINE	/MISC/	A LINE OF DATA WHICH SERVES AS A TEMPORARY STORAGE FOR A PRINTED LINE OF OUTPUT
LIOC	RDMISS	ARRAY OF IOC DATES FROM SCHEDULE CARD IN RDMISS
LL	CNTRPT	LOCATION OF DATA FOR THIS CONTAINER IN CARGO ELEMENT TABLE
LL	CSTRPT	INDEX VARIABLE
~~LL~~	SORT	~~ELEMENT NO. ON WHICH ROUTINE SORTS THE COLUMN~~
LNAME	RDMISS	CONTAINS CARGO NAME WHILE PROCESSING SCHEDULE AND SATELLITE CARDS.
LNDEX	SPRINT	VALUE OF INDEX FOR LAST CARGO ITEM
~~LNGTH~~	~~/MISS/~~	~~NUMBER OF CARGO ITEMS IN PHASE I CARGO TABLE~~
LNTH	MERGE	LENGTH OF COLUMN OF MATRIX FILE (510 UNLESS OTHERWISE CHANGED)
LOADFC	LEGPRO	LOAD FACTOR
~~LONGSH~~	~~RDLEG~~	~~YES OR NO INDICATING LONGSHORING CAPABILITY~~ OR SINGLE, INDICATING ONLY ONE CARGO ITEM PER FLIGHT.
LOWCOR	/MISC/	NUMBER OF CELLS IN DDB ARRAY (SET IN DORCA)
~~LS~~	LEGPRO	~~LONGSHORING FLAG~~
LSCHD	RDMISS	ARRAY OF FLIGHTS/YEAR FOR SCHEDULING. TEMPORARY STORAGE WHEN SATELLITE CARD IS INPUT.
~~LTH~~	~~SPRINT~~	~~CUTOFF VALUE OF YEARS TO PRINT IN HEADING~~
LVEH	LEGPRO	DEFAULT VEHICLE FOR NEXT LOWER LEG
LVT	TRAFIC	LOCATION OF START OF INPUT DATA FOR THIS VEHICLE IN DDB
~~LX~~	~~CSTRPT~~	~~LOOP VARIABLE (1 - FACILITY COSTS, 2 - VEHICLE OPERATIONS COSTS)~~
LX	LEGPRO	LOCATION OF CARGO WEIGHT FOR CURRENT DIRECTION
LX	RDCRG	TEMPORARY VARIABLE
~~LYEAR~~	~~/YEAR/~~	~~LAST YEAR THAT ANY CARGO IS SHIPPED, NORMALIZED TO RLDATE. SEE FYEAR.~~
L1	CSTRPT	LOCATION OF SPREAD FUNCTION FOR VEHICLE/FACILITY DEVELOPMENT COSTS

L1	LEGPRO	LEG NAME, FIRST PART
L2	CSTRPT	LOCATION OF SPREAD FUNCTION FOR VEHICLE/FACILITY PRODUCTION COST
L2	LEGPRO	LEG NAME, LAST PART
L2WOT	/MISC/	LOGICAL TAPE NUMBER FOR WRITE-OUT OF LEVEL II CARGO TABLE
L3	LEGPRO	NAME OF NEXT LOWER LEG (FIRST 6 CHARACTERS)
L3	RDMISS	SCRATCH VARIABLE
L4	LEGPRO	NAME OF NEXT LOWER LEG (LAST 4 CHARACTERS)
M	CNTRPT	RELATIVE YEAR + 3. INDEX TO LINE ARRAY.
M	CSTRPT	LOC. OF FACILITY INFO TO BE CURRENTLY USED
M	FACNUM	LOC. OF FACILITY INFO IN THE DDB ARRAY BEING CONSIDERED
M	LEGPRO	INDEX VARIABLE
M	MERGE	NUMBER OF MATRIX COLUMNS TO READ FROM SCRATCH FILE.
M	PERLNK	M = 1 FOR TOTALLY EXPENDABLE VEHICLE, M = 2 FOR EXPENDABLE UPPER STAGE ONLY, M = 3 FOR TOTALLY REUSABLE VEHICLE.
M	PROLNK	M = 1 FOR TOTALLY EXPENDABLE VEHICLE, M = 2 FOR EXPENDABLE UPPER STAGE ONLY, M = 3 FOR TOTALLY REUSABLE VEHICLE.
M	PROPCL	M = 1 FOR TOTALLY EXPENDABLE VEHICLE, M = 2 FOR EXPENDABLE UPPER STAGE ONLY, M = 3 FOR TOTALLY REUSABLE VEHICLE.
M	RDMISS	SCRATCH VARIABLE
M	RDVEH	FLAG INDICATING EXPENDABLE/REUSABLE CHARACTERISTICS OF VEHICLE. SEE PERLINK. ALSO USED AS A LOOP VARIABLE
M	SORT	INDEX VARIABLE
M	SPDAP	TEMPORARY VARIABLE
MANNED	/ASDAT/	INDICATES MANNED CAPSULE ASSIGNED THIS FLIGHT - 1, UP;2, DOWN 0, IF NO MANNED CAP. ASSIGNED
MAX	RDMISS	MAX HOLDS A CONSTANT EQUAL TO THE NUMBER OF ENTRIES IN THE PROGRAM NAME TABLE.
MAXA	FIND	MAX. ASSIGN. WHICH CAN BE LISTED IN FLTA MATRIX FOR LEG/VEH/YEAR
MAXA	/MAXS/	MAXIMUM NUMBER OF ENTRIES IN FLTA MATRIX (ASSIGNED ITEMS)
MAXC	FIND	MAX ALLOWABLE CONTIANERS FOR LEG/VEH/YEAR
MAXC	/MAXS/	MAXIMUM NUMBER OF ENTRIES IN CR LIST (REQUISITION CONTAINERS)
MAXCGN	/MISC/	TOTAL NUMBER OF COMPOSITE CARGO GROUPS
MAXF	ASINER	MAX. NO. OF FLIGHTS WHICH CAN BE TOTALED IN TW MATRIX
MAXF	/MAXS/	MAXIMUM NUMBER OF FLIGHTS FOR THIS VEHICLE, LEG AND YEAR
MAXI	ASINER	MAX. NO. OF CARGO ITEMS WHICH CAN BE STORED IN WS MATRIX
MAXI	/MAXS/	MAXIMUM NUMBER OF ENTRIES IN WS MATRIX (UNASSIGNED ITEMS)
MAXPY	TRAFIC	MAXIMUM FLIGHTS/YEAR FOR THIS TYPE VEHICLE
MAXPYR	RDVEH	MAX NO. FLIGHTS PER YEAR EQUAL TO CARD(5), SPREAD, UPWT
MAXVEH	TRAFIC	MAXIMUM NUMBER OF PHYSICAL VEHICLES OF ANY TYPE WHICH PROGRAM CAN ACCOMODATE (100)
MBASE	ASINER	MBASE = 2 IF ASINER HAS ASSIGNED ALL REGULAR CARGO AND IS NOW FILLING UP THE LAST FLIGHT WITH CARGO FROM THE CARGO BIN,

		MBASE = 3 IF SOME REGULAR CARGO STILL REMAINS UNASSIGNED.
MBUFF	MERGE	ARRAY CONTAINING DATA MERGED FROM COLUMNS OF MATRIX FILE.
MCBULK	/ASDAT/	IF NONZERO, INDICATES THAT A CREW CAPSULE WAS ASSIGNED TO THIS FLIGHT AND
		THAT BULK CARGO HAS NOT YET BEEN LOADED INTO THE CAPSULE FREE SPACE.
MCHG	/ASDAT/	IF NONZERO, INDICATES THAT THE REMAINDER OF A BULK CARGO ITEM (OF WHICH A
		PORTION WAS JUST ASSIGNED) WAS REMOVED FROM THE CAPTURE BIN AND
		REDESIGNATED AS REGULAR CARGO.
MCT	/ASDAT/	INDEX IN CD AND TBCONT TABLE INDICATING WHAT KIND OF BULK CARGO
		IS TO BE LOADED, IF MORE THAN ONE KIND EXIST
MDDB	/DDBS/	NUMBER OF LEVEL 2 CARGO ITEMS REMAINING ON TAPE
MIFA	/IVA/	MAXIMUM LENGTH OF IFA ARRAY
MILLE	RDLEG	VARIABLE WITH INTEGER NAME BUT CONTAINING THE VALUE 1000. IN FLOATING POINT
MINLD	RDVEH	MIN WEIGHT TO JUSTIFY FLIGHT
		EQUAL TO CARD(1), EXPWT, REFURB
MINWD	MERGE	DATA ELEMENT WHICH IS MINIMUM SO FAR.
MIVA	/IVA/	MAXIMUM LENGTH OF IVA ARRAY
MM	RDMISS	CARGO CATEGORY
MM	SORT	NO. OF ROWS IN MATRIX A
MN	RDMISS	MISSION NUMBER FOR THIS CARGO
MNAME	/PROG/	MISSION NAME TABLE
MNYRS	TRAFIC	LIFETIME (YEARS) OF THIS TYPE VEHICLE
MODE	/ASDAT/	CLASSIFICATION OF CARGO ITEMS ACCORDING TO TREATMENT
		MODE = 1 - CARGO REQUIRES EXPENDED VEHICLE.
		2 - CARGO IS FROM CAPTURE BIN.
		3 - CARGO IS REGULAR ITEM.
MODE	/SRTMOD/	DETERMINES WHETHER SORT IS ALPHANUMERIC (MODE=0) OR ALGEBRAIC (MODE≠0)
MONE	RDFAC	EQUAL TO -1.
MOSC	DDBSFT	NUMBER OF EMPTY CELLS IN DDB BETWEEN END OF CARGO ELEMENT DATA AND
		BEGINNING OF LEVEL I CARGO TABLE.
MOSF	DDBSFT	NUMBER OF EMPTY CELLS IN DDB BETWEEN END OF FACILITY TABLE AND BEGINNING OF
		CARGO ELEMENT TABLE
MP	MERGE	POINTER TO NEXT POSITION IN MERGE BUFFER MBUFF TO BE FILLED.
MTBL	/FTBL/	MAXIMUM NUMBER OF ENTRIES IN NFTBL ARRAY
MTD	FIND	PACKED WORD CONTAINING MODE, TYPE AND DIRECTION OF DESIRED CARGO
MTF	TRAFIC	MAXIMUM TOTAL FLIGHTS IN LIFETIME OF THIS TYPE VEHICLE
MTT	TRAFIC	MASTER TRAFFIC TABLE. INITIALLY CONTAINS NUMBER OF FLIGHTS AND NUMBER
		OF EXPENDED FLIGHTS FOR EACH TYPE VEHICLE IN EACH YEAR. LATER, THE
		EXPENDED COUNTS ARE REMOVED.
MTT	/ASDAT/	NAME OF THE MASTER TRAFFIC TABLE
MUST	RDMISS	MUST HOLDS A FLAG TO DETERMINE IF A MISSION ENTRY FOLLOWS A PROGRAM
		ENTRY IN THE INPUT DATA.
MWPL	/MISC/	NUMBER OF WORDS/LINE ON LAST PAGE OF COST REPORT (PAGE NR)
MXVH1	TRAFIC	UPPER INDEX FOR DO-LOOP PURGING A RETIRED OR EXPENDED VEHICLE

M1	RDMISS	FIRST CELL IN CE CONTAINING INDEX FOR COMPONENTS OF THIS COUPLED ITEM
M2	RDMISS	LAST CELL IN CE CONTAINING INDEX FOR COMPONENTS OF THIS COUPLED ITEM
N	ASINER	SCRATCH VARIABLE IDICATING NO. OF CELLS IN DDB USED FOR DATA FOR CURRENT VEHICLE
N	CNTRPT	RELATIVE YEAR + 3. INDEX TO LINE ARRAY.
N	CSTRPT	NO. OF FACILITY BEING CONSIDERED
N	FACNUM	CARGO ELEMENT NO. BEING CONSIDERED
N	MERGE	NUMBER OF MATRIX COLUMNS TO BE READ FROM INPUT TAPE.
N	PROPCL	NUMBER OF CURRENT STAGE
N	RDMISS	SCRATCH VARIABLE
N	SORT	EQUAL TO LL
N	SPRINT	LENGTH OF DATA FOR THIS VEHICLE IN DDB
N	TRAFIC	CARGO ELEMENT TABLE INDEX CORRESPONDING TO VEHICLE INDEX IVEH
NAFAC	/MISC/	INDICATES TO SUBROUTINES RDCRG AND RDFAC HOW TO PROCESS INPUT CARDS. NAFAC=0 - PROCESS ALL CARDS UNTIL A *TABLE* CARD IS ENCOUNTERED. NAFAC≠0 - PROCESS ONLY ONE CARD, THEN RETURN TO CALLING ROUTINE. (1-FACILITY CARD, 2-CARGO ELEMENT, 3-SCHEDULE, 4-SATELLITE)
NAME	FINDSP	NAME OF SPREAD FUNCTION (A6, A4 FORMAT)
NAME	INPRO	CONTENTS OF COLUMNS 1-10 OF INPUT CARD, USUALLY THE WORD TABLE.
NAME	RDCONT	UNIQUE CONTAINER NAME AS READ FROM CARD INPUT EQUIVALENT TO CARD(1)
NAME	RDCRG	CONTAINS THE NAME OF THE CARGO ELEMENT TO BE USED LATER IN THE MISSION DATA. EQUIVALENT TO CARD(1)
NAME	RDLEG	UNIQUE LEG NAME READ FROM CARD INPUT EQUIVALENT TO CARD(1)
NAME	RDMISS	NAME IS AN ARRAY WHICH HOLDS THE MISSION DATA CARD IMAGES. EQUIVALENT TO CARD(1)
NAME	RDSPD	NAME OF SPECIFIC SPREADING FUNCTION EQUIVALENT TO CARD(1) AND BLANK
NAME	RDVEH	VEHICLE NAME.
NASS	/ASDAT/	NO. OF ASSIGNMENTS STORED IN FLTA FOR THIS VEHICLE/LEG/YEAR
NB	CSTRPT	POINTER TO BEGINNING OF CARGO BLOCK IN DDB FOR THIS PROGRAM, MISSION AND VEHICLE.
NB	FIND	SCRATCH VARIABLE INDICATING BIT POSITION FOR PACKING AND UNPACKING DATA IN CR ARRAY.
NB	TABLES	BEGINNING OF DATA FOR THIS SPREAD FUNCTION IN DDB
NB	VEHRPT	POINTS TO THE BEGINNING LOCATION OF A SET OF DATA FOR A VEHICLE IN THE CARGO TABLE.
NBASE	LEGPRO	LOCATION OF START OF DATA FOR THIS CARGO ELEMENT
NBASE	RDMISS	NBASE POINTS TO A PROGRAM NAME IN THE PROGRAM NAME TABLE.
NBASE	TRAFIC	LOCATION OF DATA IN CARGO ELEMENT TABLE
NBCE	/CRGS/	POINTS TO THE BEGINNING OF THE CARGO ELEMENT TABLE IN THE DYNAMIC

		DATA BLOCK.
NBDDB	/DDBS/	POINTS TO THE BEGINNING OF THE CARGO TABLE IN THE DYNAMIC DATA BLOCK.
NBFAC	/FACS/	LOCATION, IN THE DYNAMIC DATA BLOCK, WHERE THE FACILITY TABLE BEGINS.
NBITS	PACK	NO. OF BITS HANDLED IN PACK
NBLK	/DDBS/	NUMBER OF 510-WORD BLOCKS OF CELLS AVAILABLE IN DDB TO READ IN LEVEL 2 DATA
NBMISS	/MISS/	LOCATION OF BEGINNING OF PHASE I CARGO TABLE IN DDB.
NBSPD	/SPDS/	POINT TO THE BEGINNING LOCATION OF THE SPREAD TABLE IN THE DYNAMIC DATA BLOCK.
NBVEH	PERLNK	START OF DATA FOR THIS VEHICLE IN DDB
NBVEH	/VEHS/	LOCATION OF FIRST ENTRY OF VEHICLE TABLE IN DDB
NC	SPRINT	LOOP VARIABLE FOR NUMBER OF PAGES (DEPENDS ON YEAR SPAN)
NCALL	/ASDAT/	NUMBER OF CONSECUTIVE UNSUCCESSFUL CALLS TO SUBROUTINE -FIND-
NCARD	RDFAC	NO. OF CARDS READ
NCE	/CRGS/	NUMBER OF CARGO ELEMENTS IN THE CARGO ELEMENT TABLE.
NCEL	FACRPT	CARGO ELEMENT NO. OF LAST FACILITY CONSIDERED BEFORE NCE
NCEP	CSTRPT	CARGO ELEMET NO.
NCEP	FACRPT	CARGO ELEMET NO. OF THE PRESENTLY CONSIDERED FACILITY
NCGN	LEGPRO	COMPOSITE GROUP NUMBER
NCNT	FACNUM	NO. OF FACILITIES SHIPPED THAT YEAR
NCOLS	MERGE	NUMBER OF COLUMNS OF INPUT MATRIX STILL REMAINING ON INPUT TAPE.
NCONT	/CONT/	NUMBER OF CONTAINERS INPUT
NCPR	MERGE	NUMBER OF COLUMNS OF INPUT MATRIX PER FULL LOGICAL RECORD ON INPUT TAPE.
NCR	/ASDAT/	NUMBER OF ENTRIES IN CONTAINER REQUISITION LIST CR.
NCRD	RDCONT	NO. OF CARDS READ EQUAL TO NCONT
NCRD	RDCRG	NO. LAST READ CARD OF CARGO ELEMENT TABLE EQUIVALENCED TO NCE
NCRD	RDFAC	NUMBER OF FACILITY CARDS READ (EQUIVALENCED TO NCARD)
NCRD	RDLEG	NO. OF CARDS READ EQUAL TO NLEG
NCRD	RDSPD	NO. OF CARDS READ BY THIS SUBROUTINE
NCRD	RDVEH	NO. OF CARDS READ AS INPUT
NCTMAX	/CONT/	MAXIMUM NUMBER OF CONTAINERS WHICH CAN BE INPUT
NDDB	/DDBS/	NUMBER OF CARGO SHIPMENTS
NDEX	CNTRPT	IF NONZERO, THIS CONTAINER WAS ALSO INPUT TO FACILITIES TABLE AND IS TO BE ADDED TO IFA TABLE FOR COSTING.
NDEX	LEGPRO	CONTAINER NO. THAT IS BEING SHIPPED
NDEX1	LEGPRO	CURRENT INDEX OF LEG/VEH/YEAR
NDEX2	LEGPRO	NEXT INDEX OF LEG/VEH/YEAR
NDEX3	LEGPRO	LAST LEG/YR
NDEX4	LEGPRO	CURRENT LEG/YR
NDEX5	LEGPRO	NEXT LEG/YR
NDP	VALUE	NO. OF DECIMAL POINTS

NDXCE	LEGPRO	INDEX OF CARGO ELEMENT TABLE
NEV	ASINER	NUMBER OF EXPENDED VEHICLES REQUESTED FOR THIS LEG/YEAR.
~~NEWLEG~~	~~LEGPRO~~	~~INDEX OF LOWER LEG FOR THIS CARGO GROUP~~
NEWLEG	TRAFIC	LEG OF CURRENT INTEREST, IF TRAFIC WAS CALLED FROM LEGPRO. NOT USED WHEN TRAFIC IS CALLED FROM REPORT.
~~NEXP~~	~~PROLNK~~	~~EQUALS 1 IF VEHICLE IS TO BE EXPENDED.~~
NEXP	/ASDAT/	NO. OF UNASSIGNED CARGO ITEMS WHICH REQUIRE EXPENDABLE VEHICLE
NEXT	READER	INDICATES WHERE TO START SHIFT OF CARD COLUMNS WHEN REFORMATTING INPUT CARDS WHICH CONTAIN = SIGNS
NEXTRA	TRAFIC	NUMBER OF EXTRA FLIGHTS IN LIFETIME OF THIS VEHICLE WHICH WILL REMAIN UNUSED IF VEHICLE IS USED AT ITS MAXIMUM RATE IN ITS REMAINING YEARS ~~AFTER CURRENT YEAR.~~
NE1	TRAFIC	NUMBER OF EXPENDED FLIGHTS
NF	LEGPRO	EQUALS NIFA
NF	VALUE	POSITION OF FIRST NON-BLANK CHARACTER
NFAC	/FACS/	NUMBER OF FACILITY ENTRIES IN THE FACILITY TABLE.
NFEXP	TRAFIC	NUMBER OF FLIGHTS REMAINING FOR A VEHICLE TO BE EXPENDED
~~NFILE~~	~~/SEG32/~~	~~NUMBER OF 510-WORD RECORDS WHICH CAN BE ACCOMODATED IN AVAILABLE SPACE~~
NFLAG	RDVEH	NFLAG = 4 FOR DRY STAGES, = 5 FOR WET STAGES CARD.
NFLT	/ASDAT/	NO. FLIGHTS PER VEHICLE PER LEG PER YEAR
NFLTS	TRAFIC	NUMBER OF FLIGHTS STILL UNASSIGNED IN THIS YEAR FOR THIS TYPE VEHICLE
NFR	TRAFIC	ARRAY CONTAINING NUMBER OF FLIGHTS REMAINING IN LIFETIME OF EACH VEHICLE ENTERED IN DVTT.
~~NFR~~	~~VEHRPT~~	~~EQUIVALENCE TO NFTBL~~
NFR	/ASDAT/	NUMBER OF FLIGHTS REMAINING FOR A VEHICLE IN A FLEET OF A GIVEN TYPE OF VEHICLES.
~~NFTBA~~	~~TRAFIC~~	~~NUMBER OF FLIGHTS TO BE ASSIGNED TO CURRENT VEHICLE~~
NFTBL	/FTBL/	FLIGHT TABLE. EACH ENTRY IS A PACKED WORD CONTAINING VEHICLE NUMBER, YEAR, NUMBER OF FLIGHTS, AND NUMBER OF EXPENDED VEHICLES.
~~NFW~~	~~/MISC/~~	~~FIRST ELEMENT OF INTEREST IN COST1, ETC., ARRAYS~~
NF1	TRAFIC	NUMBER OF FLIGHTS EXTRACTED FROM NFTBL ARRAY. ALSO, NUMBER OF ACQUIRED VEHICLES LIST IN VEHICLE ACQUISITION TABLE
~~NIF~~	~~CSTRPT~~	~~NUMBER OF FACILITIES ENTERED IN IFAC ARRAY.~~
NIFA	/IFA/	NUMBER OF ENTRIES IN FACILITY ACQUISITION LIST
NIOC	RDMISS	TEMPORARY VARIABLE FOR IOC DATE
~~NIVA~~	~~/IVA/~~	~~NUMBER OF ENTRIES IN VEHICLE ACQUISITION LIST~~
NL	CSTRPT	POINTER TO END OF CARGO BLOCK IN DDB FOR THIS PROGRAM, MISSION, AND VEHICLE.
~~NL~~	~~TABLES~~	~~END OF DATA FOR THIS SPREAD FUNCTION IN DDB~~
NL	VALUE	POSITION OF LAST NON-BLANK CHARACTER
NL	VEHRPT	POINTS TO THE LAST LOCATION OF THE SET OF DATA FOR A VEHICLE IN THE ~~DYNAMIC DATA BLOCK.~~
NLCE	/CRGS/	POINTS TO END OF CARGO ELEMENT TABLE IN DDB.

NLDDB	/DDBS/	LOCATION OF END OF PHASE II CARGO TABLE IN DDB.
NLEG	/LEGS/	NO. OF ENTRIES IN LEG TABLE
NLFAC	/FACS/	LOCATION OF END OF FACILITY DATA IN DDB ARRAY.
NLGMAX	/LEGS/	MAX. DIMENSION OF THE LEG TABLE SET TO 31 IN DORCA
NLMISS	/MISS/	LOCATION OF END OF PHASE I CARGO TABLE IN DDB
NLSPD	/SPDS/	POINTS TO THE LAST ENTRY OF THE SPREAD TABLE IN THE DYNAMIC DATA BLOCK.
NLVEH	/VEHS/	POINTS TO THE LAST ENTRY OF THE VEHICLE TABLE
NLW	/MISC/	LAST ELEMENT OF INTEREST IN COST1, ETC., ARRAYS
NL1	TRAFIC	LEG INDEX OF ACQUIRED VEHICLE(S)
NM	CSTRPT	MISSION NO.
NM	FACRPT	MISSION NO. OF FACILITY CURRENTLY BEING CONSIDERED
NMISS	/PROG/	NMISS HOLDS THE NUMBER OF ENTRIES IN THE MISSION NAME TABLE.
NML	FACRPT	MISSION NO. OF LAST FACILITY CONSIDERED BEFORE NM
NMODE	/ASDAT/	NMODE(I) = NUMBER OF CARGO ITEMS OF MODE I STILL UNASSIGNED. I=1 FOR CARGO REQUIRING AN EXPENDED VEHICLE I=2 FOR ITEMS IN THE CAPTURE BIN I=3 FOR REGULAR CARGO ITEMS
NN	RDMISS	VEHICLE INDEX INSERTED INTO IVA TABLE ENTRY
NN	SORT	NO. OF COLUMNS IN MATRIX A
NO	ASINER	CONTAINS THE CODED WORD --NO-- FOR COMPARISON
NOBOX	LEGPRO	USED TO CREATE UNIQUE SYMBOL FOR NAME OF NEW COUPLED CARGO BOX
NOCC	/ASDAT/	NOCC(I) IS THE NUMBER OF ITEMS ALREADY ASSIGNED TO CURRENT FLIGHT UPWARDS (I=1) AND DOWNWARDS (I=2). A CONTAINER PLUS ALL ITS CONTENTS COUNTS AS A SINGLE ITEM FOR THIS PURPOSE.
NOFLT	LEGPRO	TOTAL NUMBER OF FLIGHTS OF VEHICLE = VEHNDX ON CURRENT LEG, INCLUDING ANY RESULTING FROM PREVIOUS CALLS TO ASINER
NOSC	/MISC/	NUMBER OF CELLS NEEDED IN DDB TO STORE 10 MORE CARGO ELEMENTS.
NOSF	/MISC/	NUMBER OF CELLS NEEDED IN DDB TO STORE DATA FOR 10 MORE FACILITIES.
NOVEH	/VEHS/	NUMBER OF ACTIVE VEHICLES IN FLEET. COMPUTED IN LEGPRO, USED IN SPRINT.
NP	FACRPT	PROG. NO. OF FACILITY CURRENTLY BEING CONSIDERED
NPF	LEGPRO	NUMBER OF PROPELLANT TANKS NEEDED TO FUEL FLIGHTS SCHEDULED FOR THIS LEG, VEHICLE AND YEAR.
NPL	FACRPT	PROG. NO. OF LAST FACILITY CONSIDERED BEFORE NP
NPM	CSTRPT	PROGRAM/MISSION CORRESPONDING TO DATA STORED IN ARRAY LINE.
NPMCE	FACNUM	PACKED PROGRAM, MISSION AND CARGO ELEMENT NO.S BEING CONSIDERED
NPMCEL	FACNUM	PACKED PROG., MISSION AND CARGO ELE. NO.S PREVIOUSLY CONSIDERED
NPREF	/VEHS/	NUMBER OF VEHICLES INPUT TO VEHICLE PREFERENCE LIST
NPRI	/ASDAT/	IF NPRI=1, DEBUGGING INFORMATION OF THE ASSIGNMENT PROCEDURE IS PRINTED IN SUBROUTINES ASINER AND FIND.
NPROG	/PROG/	NPROG HOLDS THE NUMBER OF ENTRIES IN THE PROGRAM NAME TABLE.
NPU	PROLNK	NUMBER OF PROPULSIVE (WET) STAGES
NR	SORT	EQUAL TO MM

NR	/MISC/	NUMBER OF PAGES REQUIRED FOR EACH COST REPORT (ALSO, NUMBER OF SCRATCH TAPES)
NREAD	MERGE	NUMBER OF LOGICAL RECORDS TO READ FROM INPUT FILE INTO FILE MATRIX.
NREQ	PERLNK	NUMBER OF STAGES REQUIRED. SHOULD EQUAL NTOT.
NREQ	PROLNK	NUMBER OF STAGES REQUIRED
NREQ	PROPCL	NUMBER OF STAGES REQUIRED. SHOULD EQUAL NTOT.
NROWS	MERGE	NUMBER OF ROWS IN INPUT MATRIX. SAME AS SUBROUTINE ARGUMENT NR.
NRV	TRAFIC	NUMBER OF REQUIRED/REMAINING VEHICLES FOR THIS YEAR
NSCD	RDMISS	NUMBER OF SCHEDULE DATES STORED IN THE LTOC ARRAY
NSCHD	RDMISS	CONTAINS CODED WORD -SCHEDU- OR -SHIPMT- WHILE RDMISS IS PROCESSING SCHEDULES
NSFT	DDBSFT	NUMBER OF CELLS TO SHIFT CARGO TABLE.
NSHP	RDMISS	NUMBER OF VALUES OF SHIPMENT DATA STORED IN THE LSCHD ARRAY
NSPD	/SPDS/	NUMBER OF SPREAD FUNCTIONS
NSW	READER	FLAG SET TO ONE WHEN INPUT CARDS ARE TO BE PRINTED
NTBL	/FTBL/	NO. OF ITEMS IN NFTBL TABLE
NTBL1	/FTBL/	LOWER INDEX OF SUBSET OF NFTBL TO BE PROCESSED BY SUBROUTINE -TRAFIC-
NTBL2	/FTBL/	UPPER INDEX OF SUBSET OF NFTBL TO BE PROCESSED BY SUBROUTINE -TRAFIC-
NTOT	PERLNK	NUMBER OF WET STAGES IN THIS VEHICLE
NTOT	PROPCL	NUMBER OF STAGES AVAILABLE TO VEHICLE
NTW	CSTRPT	NUMBER OF YEARS IN THE TIME PERIOD WHICH WILL BE PRINTED
NTYRS	/YEAR/	ARRAY CONTAINING YEARS THAT CONTAINERS ARE USED
NUMBER	RDMISS	MULTIPLICITY OF CARGO ELEMENT (NUMBER OF UNITS SHIPPED)
NV	CSTRPT	POINTER TO BEGINNING OF DATA FOR THIS VEHICLE IN DDB.
NV	LEGPRO	EQUALS NIVA
NV	PERLNK	INDEX OF THIS WET STAGE IN VEHICLE TABLE.
NV	VEHRPT	NUMBER OF VEHICLES IN CARGO TABLE
NVEH	/VEHS/	NUMBER OF VEHICLES IN THE VEHICLE TABLE
NVHMAX	/VEHS/	MAX. NO. OF VEHICLES ALLOWED IN THE VEHICLE TABLE. SET TO 30 IN DORCA
NVIF	TRAFIC	NUMBER OF VEHICLES IN FLEET IN CURRENT YEAR
NVMAX	TRAFIC	POINTER TO LAST ENTRY IN MTT AND FLTAC MATRICES
NVMAX	/MISC/	INDEX OF LAST VEHICLE ACTUALLY USED. NVMAX ≤ NVEH.
NV1	TRAFIC	VEHICLE INDEX OF ENTRY IN IVA/NFTBL ARRAYS
NWCE	/CRGS/	NWCE HOLDS THE NUMBER OF WORDS PER CARGO ELEMENT TABLE ENTRY.
NWDDB	/DDBS/	NUMBER OF WORDS PER ITEM IN PHASE II CARGO TABLE (EQUALS 3).
NWFAC	/FACS/	NUMBER OF WORDS FOR EACH FACILITY ENTRY.
NWMISS	/MISS/	NWMISS HOLDS THE NUMBER OF WORDS PER MISSION DATA ENTRY.
NWORD	RDMISS	NWORD HOLDS THE CODED LIST OF VEHICLES SHIPPED TO BE PUT INTO THE IVA ARRAY.
NWORDS	MERGE	NUMBER OF WORDS TO READ FROM INPUT OR SCRATCH TAPE IN CURRENT LOGICAL RECORD
NWPL	/MISC/	NUMBER OF WORDS/LINE ON COST REPORT, PAGE 1,2,...,NR-1.
NWPR	MERGE	NUMBER OF WORDS IN A FULL LOGICAL RECORD ON INPUT FILE.
NWT	LEGPRO	WEIGHT LOAD FACTOR FOR VEHICLE, X 100000 (INTEGER)

NWVEH	/VEHS/	NO. OF WORDS IN THE BASIC VEHICLE TABLE EQUAL TO 16 EXCLUDING LEG DATA
NXTLEG	LEGPRO	INDEX OF LOWER LEG FOR NEXT CARGO GROUP
NXTWRD	RDCRG	POINTS TO THE NEXT WORD IN THE DYNAMIC DATA BLOCK WHERE NEXT ENTRY WILL BE STORED.
NXTWRD	RDSPD	NEXT WORD IN THE DYNAMIC DATA BLOCK IN WHICH THE NEXT ENTRY WILL BE STORED
NXTWRD	RDVEH	INCREMENTING INTEGER USED IN INDEX
NYA	TRAFIC	YEAR IN WHICH THIS VEHICLE WAS ACQUIRED
NYACT	TRAFIC	NUMBER OF YEARS THIS VEHICLE HAS BEEN ACTIVE, INCLUDING CURRENT YEAR
NYEAR	SPRINT	FIRST YEAR OF ENTIRE CASE.
NYLEFT	TRAFIC	NUMBER OF YEARS REMAINING IN LIFETIME OF THIS VEHICLE AFTER THIS YEAR.
NYR	CSTRPT	YEAR IN WHICH FACILITY IS USED
NYR	FACNUM	YEAR IN WHICH THE PROG./MISS/CE TAKES PLACE
NYR	RDSPD	NUMBER OF YEARS OVER WHICH SPREADING FACTORS APPLY. EQUIVALENT TO CARD(3) AND FACTOR
NYRS	/YEAR/	NUMBER OF YEARS SPANNED BY ALL MISSIONS.
NYRS1	/YEAR/	1 + NUMBER OF YEARS SPANNED BY ALL MISSIONS. NYRS1 = NYRS+1
NYRS3	/YEAR/	3 + NUMBER OF YEARS SPANNED BY ALL MISSIONS. NYRS3 = NYRS+3
NY1	TRAFIC	DATE OF ENTRY IN NFTBL/IVA ARRAYS
N1	CNTRPT	REPRENTS BOTH YEAR AND COUNT FOR CONTAINER ENTRY TO IFA TABLE
N1	VALUE	N1 = NL - 9 INDEX VARIABLE
ONE	FINDSP	DEFAULT VALUE FOR SPREAD TABLE POINTER IPT
ONE	RDCONT	CONSTANT OF VALUE 1.
ONE	RDLEG	CONSTANT SET TO 1.
OPTN	READER	LIST OF ACCEPTABLE OPTION NAMES
OPTV	READER	LIST OF ACCEPTABLE HOLLERITH VALUES FOR EACH OPTION
OUT	MERGE	INDEX OF OUTPUT TAPE FOR CURRENT MERGE.
OUTAPE	MERGE	LOGICAL NUMBER OF OUTPUT TAPE FOR CURRENT MERGE.
P	MERGE	P(I,J) CONTAININS POINTERS CONCERNING COLUMN J OF MATRIX FILE.
P	VALUE	VALUE IN CODED FORMAT BUT RIGHT-ADJUSTED IN FIELD OF 10 CHARACTERS.
PENAL	RDCONT	CONTAINER EMPTY WEIGHT.
PHASE	RDMISS	PHASE HOLDS THE PHASE OF OPERATION ENTRY IN MISSION DATA.
PKWRD	RDCRG	DUMMY VARIABLE USED FOR TEMPORARY STORAGE
PL	PERLNK	ASSUMES VARIOUS VALUES OF PAYLOAD WEIGHT DURING THE ITERATION FOR WPREQ
PLL	PERLNK	VALUE OF PL TWO ITERATIONS AGO
PLMX	PERLNK	TEMPORARY STORAGE FOR WPLUP, WHICH IS CHANGED WHILE WPLON IS BEING COMPUTED
PLN	PERLNK	VALUE OF PL ON LAST ITERATION
PN	RDMISS	PROGRAM NUMBER FOR THIS CARGO (INTEGER)
PNAME	/PROG/	PNAME ARRAY HOLDS MISSION NAMES PROVIDED BY MISSION DATA INPUT.
PNMN	LEGPRO	PROG/MISSION COMBINATION
PNMN	RDMISS	ARRAY WITH COMPOSITE PROGRAM/MISSION NUMBER (6 BITS EACH)
PNTER	RDCRG	POINTER TO EITHER THE VEHICLE OR FACILITIES TABLE

C-3

PROP	LEGPRO	PACKED FIRST WORD FOR PROPELLANT TANK ADDED TO PHASE I CARGO TABLE
PROP	PROLNK	TOTAL PROPELLANT AVAILABLE TO WHOLE VEHICLE (SUM OF MAXIMUM PROPELLANT FOR EACH STAGE)
PROPEL	RDVEH	PROPELLANT WEIGHT FOR COMPLETE FUELING OF VEHICLE EQUAL TO CARD(3),NREV,LEGNM
PROPOL	LEGPRO	WEIGHT OF PROPELLANT OFF-LOADED INTO LAST STAGE OF VEHICLE
PROPOL	PROLNK	AMOUNT OF PROPELLANT OFF-LOADED IN LAST STAGE
PROPUP	/MISC/	CAPACITY OF PROPELLANT TANK
PROPWT	LEGPRO	WEIGHT OF PROPELLANT NEED TO FUEL ONE FLIGHT OF THIS VEHICLE WHEN FULLY LOADED
PROPWT	PROLNK	TOTAL PROPELLANT WEIGHT REQUIRED (COMPUTED BY PROLNK)
PROP2	LEGPRO	PACKED SECOND WORD FOR PROPELLANT TANK ADDED TO PHASE I CARGO TABLE
PW	PACK	THE PACKED WORD
R	PERLNK	IN ROCKET EQUATION, REPRESENTS RATIO OF INITIAL WEIGHT TO END WEIGHT DURING BURNING OF STAGE.
RELDT	LEGPRO	DATE CARGO ITEM IS TO BE SHIPPED
RFLT	RDVEH	EQUAL TO CARD(7), LIFFLT(1) AND RPROD(1) NOT IN USE
RLDATE	/YEAR/	THE RELATIVE YEAR TO WHICH THE TIME IS BASED. I.E. 1970
RPROD	RDFAC	RECURRING PRODUCTION COST
RPROD	RDVEH	RECURRING PRODUCTION COST.
RVOL	/ASDAT/	REMAINING VEHICLE VOLUME CAPACITY ON CURRENT FLIGHT IN UP DIRECTION (I=1) AND DOWN DIRECTION (I=2).
S	PROPCL	≠SAFETY≠ FACTOR CONCERNING ENGINE THRUST. EQUALS 1 IF INPUT PERFORMANCE DATA IS PRACTICAL, > 1 IF INPUT DATA IS THEORETICAL. ALWAYS = 1 IN DORCA.
SPD	RDSPD	CONTAINS SUM OF SPREADING FACTORS.
SPPROD	RDVEH	NAME OF SPREAD FUNCTION TO BE USED FOR RECURRING PRODUCTION COST. NOT REFERENCED BY NAME BUT EQUIVALENCED TO CARD(9-10).
SPREAD	RDVEH	NAME OF SPREAD FUNCTION TO BE USED FOR NONRECURRING DEVELOPMENT COST. NOT REFERENCED DIRECTLY BUT EQUIVALENCED TO CARD(5-6).
SUCESS	RDLEG	NAME OF NEXT LOWER (SUCCESSOR) LEG.
T	SORT	VARIABLE CONTAINING THE VALUE OF THE MEDIAN ELEMENT
TBCONT	/CONT/	MATRIX CONTAINING 8 WORDS OF DATA FOR EACH CONTAINER
		WORDS 1-2 CONTAINER NAME
		3 CAPACITY (LBS)
		4 CONTAINER WEIGHT (LBS)
		5 CLASSIFICATION (1-CREW, 2-BULK, 4-PROPELLANT)
		6 VOLUME
		7 USED AS A FLAG IN SUBROUTINE CNTRPT
		8 RETURN/EXPEND OPTION
TBLEG	/LEGS/	TABLE CONTAINING ALL LEG DATA
TBL1	SPRINT	LEG NAME, FIRST 6 CHARACTERS
TBL2	SPRINT	LEG NAME, LAST 4 CHARACTERS
TBSPD	/SPDS/	SPREAD TABLE NAME (LOCATION)

TBVEH	/VEHS/	MATRIX CONTAINING 4 WORDS OF INFORMATION FOR EACH VEHICLE WORDS 1-2 VEHICLE NAME 3 PACKED WORD CONTIANING START LOCATION AND LENGTH OF DATA FOR THIS VEHICLE IN THE DDB ARRAY. 4 STAGES (UP TO 6 STAGES, PACKED 6 BITS/STAGE)
TCAP	FIND	TEMPORARY VARAIBLE CONTAINING UNUSED CONTAINER CAPACITY
TDAT	MERGE	TDAT(I,J) GIVES DATA ON FILE INDEX J. J = 1,2,3 ARE SCRATCH TAPES, J = 4 IS THE INPUT TAPE.
TEMP	LEGPRO	TEMPORARY VARIABLE USED IN DATA INTERCHANGE
TEMP	RDMISS	TEMP IS A TEMPORARY BUFFER WHICH IMPLEMENTS THE FORMAT CONVERSION OF 18 CHARACTER MISSION AND PROGRAM NAMES.
TEMP	RDVEH	TEMPORARY VARIABLE USED FOR PACKING OF STAGE INDICES
TIME	LEGPRO	= 0, IF INTIAL RUN-THROUGH FOR VEH/LEG/YEAR
TLF	SPRINT	LOAD FACTOR OF CURRENT CARGO ITEM.
TLOAD	SPRINT	TOTAL OF LOAD FACTORS FOR ALL CARGO ITEMS ON THIS FLIGHT.
TOT	CNTRPT	TOTAL NO. OF CONTAINERS LEAVING EARTH PER YEAR
TOTAL	ASINER	TOTAL AMOUNT OF WHICHEVER KIND OF CARGO REMAINS IN SMALLEST SUPPLY IN SAME DIRECTION AS MANNED CAPSULE TO BE FILLED. USED TO DETERMINE WHICH KIND OF BULK TO LOAD.
TOTAL	SPRINT	CONTAINS CODED WORD -TOTAL- FOR PRINTOUT
TOTFLT	TRAFIC	TOTAL NUMBER OF VEHICLES AVAILABLE IN FLEET IN EACH YEAR
TOTFLT	/ASDAT/	FOR A GIVEN TYPE OF VEHICLE, A RUNNING TOTAL OF FLTAC BY YEAR
TOTWT	LEGPRO	TOTAL WEIGHT CARRIED BY VEHICLE UP AND DOWN ON CURRENT FLIGHT, EXPRESSED AS EQUIVALENT UP-WEIGHT.
TPROPW	LEGPRO	ARRAY GIVING TOTAL PROPELLANT REQUIRED FOR ALL FLIGHTS OF EACH VEHICLE ON THIS LEG/YEAR. TPROPW(I) = TOTAL PROPELLANT FOR VEHICLE NO. I
TRIFLF	/ASDAT/	CUTOFF VALUE TO DETERMINE WHEN CERTAIN QUANTITIES HAVE BEEN EXHAUSTED. EQUAL TO 0.999 LB.
TRLF	SPRINT	LOAD FACTOR FOR THIS FLIGHT
TT	SORT	VARIABLE USED IN INTERCHANGING COLUMNS.
TVCAP	ASINER	SCRATCH VARIABLE FOR REMAINING VEHICLE CAPACITY
TW	/ASDAT/	TW(I,J) GIVES THE TOTAL WEIGHT TO BE CARRIED BY TH VEHICLE ON FLIGHT NO. J IN DIRECTION I (1-UP, 2-DOWN)
TWDN	PROLNK	TOTAL PAYLOAD WEIGHT DOWN
TWO	RDCONT	CONSTANT OF VALUE 2.
TWTDW	SPRINT	TOTAL DOWN WEIGHT ON THIS FLIGHT.
TWTUP	SPRINT	TOTAL UP WEIGHT ON THIS FLIGHT.
TWUP	PROLNK	TOTAL PAYLOAD WEIGHT UP
TYPE	/ASDAT/	CLASSIFICATION OF CARGO ITEMS BY STORAGE REQUIREMETNTS 1- CREW, 2-BULK CARGO, 3-DISCRETE (SELF-CONTAINED) ITEM.
UP	SPRINT	UP WEIGHT OF CARGO ITEM.
UPMAX	ASINER	MAX. WEIGHT WHICH VEHICLE CAN CARRY UPWARDS, ASSUMING VEHICLE RETURNS EMPTY.

UPMAX	LEGPRO	MAXIMUM PAYLOAD UP FOR THIS VEHICLE
UPMAX	SPRINT	MAXIMUM VEHICLE WEIGHT CAPACITY UPWARDS
UPWT	RDCRG	DESCRIBES THE UP WEIGHT OF THE CARGO ELEMENT IF THE CARGO ELEMENT IS TO BE SHIPPED UP THE LEG. EQUIVALENT TO CARD(10)
UPWT	RDVEH	2-CELL ARRAY CONTAINING INPUT CODED REPRESENTATION OF UPMAX (MAXIMUM PAYLOAD VEHICLE CAN CARRY UPWARDS ON THIS LEG IF NOT EXPENDED)
UPWT	RDVEH	MAXIMUM UP WEIGHT FOR VEHICLE ON THIS LEG. NOT REFERENCED BY NAME BUT EQUIVALENCED TO CARD(5-6).
V	PACK	THE VARIABLE FROM/TO WHICH THE BITS ARE UNPACKED/PACKED
V	PROPCL	VELOCITY AVAILABLE FROM THRUST OF THIS STAGE
V	RDCONT	THE VALUE OF VOLUME IN FLOATING POINT
V	VALUE	CONVERTED VALUE IN FLOATING POINT
VCAP	/ASDAT/	AMOUNT OF VEHICLE CARRYING CAPACITY UNUSED, EXPRESSED AS EQUIVALENT UP-WEIGHT
VDATA	CSTRPT	ARRAY CONTAINING NAME (2 WDS), TOTAL COST OR COUNT FOR ALL YRS (1 WD), AND COST OR COUNT FOR EACH YEAR, 1970-1999 (30 WDS.) EQUIVALENCED TO ARRAY LINE.
VDATA	VEHRPT	CONTAINS DATA EXTRACTED FROM CARGO TABLE PERTAINING TO EACH VEHICLE.
VDONE	/ASDAT/	IF VDONE=1, CURRENT VEHICLE FLIGHT IS FILLED UP.
VEH	/VEHS/	VEH(I) = 1 IF VEHICLE I IS ACTIVE, = 0 IF NOT.
VEHICL	RDLEG	NAME OF DEFUALT VEHICLE EQUAL TO CARD(7)
VEHNDX	LEGPRO	CURRENT VEHICLE NO. IN DCB ARRAY
VEHNO	LEGPRO	NEXT VEHICLE NO. IN DDB ARRAY
VEH1	TRAFIC	FIRST OF TWO PACKED WORDS USED TO ENTER NEWLY ACQUIRED VEHICLES INTO PHASE I CARGO TABLE.
VEH2	TRAFIC	SECOND OF TWO PACKED WORDS USED TO ENTER NEWLY ACQUIRED VEHICLES INTO PHASE I CARGO TABLE.
VIN	PERLNK	VELOCITY INCREMENT DELTA V REQUIRED TO TRANSFER FROM ORBIT AT LEG BOTTOM TO ORBIT AT TOP OF LEG (INDEPENDENT OF VEHICLE)
VIN	PROPCL	VELOCITY INCREMENT DELTA V REQUIRED FOR ORBIT CHANGE (LEG TRAVEL)
VOL	/ASDAT/	VOLUME OF UNASSIGNED CARGO ITEM I THIS ARRAY ACCOMPANIES MATRIX WS AND IS INDEXED THE SAME.
VOLF	SPRINT	VOLUME FACTOR FOR THIS FLIGHT
VOLMAX	SPRINT	VEHICLE VOLUME CAPACITY
VOLMAX	/ASDAT/	MAX. VOLUME CAPACITY OF CURRENT CARGO
VOLUME	ASINER	VOLUME FACTOR OF CURRENT CARGO ITEM
VOLUME	FIND	VOLUME FACTOR OF CURRENT CARGO ITEM
VOLUMF	RDCONT	INPUT VOLUME FACTOR IN CODED FORMAT.
VOLUME	RDCRG	VOLUME FACTOR OF CURRENT CARGO ITEM
VOLUMF	SPRINT	TOTAL VOLUME ON FLIGHT
VOLV	LEGPRO	VEHICLE VOLUME CAPACITY

VOL1	LEGPRO	VOLUME CAPACITY OF CURRENT VEHICLE (VEHNDX)
VOL2	LEGPRO	VOLUME CAPACITY OF VEHICLE NO. LVEH (LOWER LEG DEFAULT VEHICLE)
VPREF	/VEHS/	VPREF(N) IS A PACKED WORD CONTAINING INFORMATION ON THE N-TH VEHICLE IN THE VEHICLE PREFERENCE LIST BITS 0-17 VEHICLE INDEX IN TBVEH 18-26 FIRST YEAR VEHICLE IS AVAILABLE (NORMALIZED TO RLDATE) 27-36 LAST YEAR VEHICLE IS AVAILABLE
VRQ	PROPCL	VELOCITY REQUIRED OF STAGES NOT YET CALCULATED
VTOT	VEHRPT	SUM OF LOAD FACTORS FOR ALL FLIGHTS, GIVEN BY VEHICLE AND YEAR
VTOT	/ASDAT/	CONTAINS SUMMARY DATA EXTRACTED FROM CARGO TABLE PERTAING TO EACH VEHICLE
VV	LEGPRO	PACKED VEHICLE INDICES FROM SECOND AND THIRD LOWER LEG (6 BITS EACH)
V1	LEGPRO	INDEX OF VEHICLE USED ON CURRENT LEG
V1	/ASDAT/	FIRST PART OF CURRENT VEHICLE NAME
V2	LEGPRO	INDEX OF VEHICLE USED ON NEXT LOWER LEG
V2	/ASDAT/	SECOND PART OF CURRENT VEHICLE NAME
V3	LEGPRO	INDEX OF VEHICLE USED ON SECOND LOWER LEG, IF ANY
V4	LEGPRO	INDEX OF VEHICLE USED ON THIRD LOWER LEG, IF ANY
W	ASINER	ITEM WEIGHT
W	FIND	ITEM WEIGHT
WB	PROPCL	WB(I) = WEIGHT OF I-TH STAGE AT BURNOUT
WBO	PERLNK	WEIGHT OF STAGE AT BURNOUT, INCLUDING NON-USABLE FUEL.
WE	PROPCL	WEIGHT OF VEHICLE AT END OF BURNOUT OF STAGE N
WEIGHT	FIND	EQUIVALENT UP-WEIGHT OF CARGO PLUS ITS CONTAINER (IF ANY) ON ROUND TRIP.
WLEFT	/ASDAT/	WLEFT(I,J,K) = TOTAL WEIGHT OF STILL-UNASSIGNED CARGO OF MODE I, TYPE J, DIRECTION K.
WLF	RDMISS	ARRAY OF COUPLED COMPONENT WEIGHT LOAD FACTORS, IN INTEGER FORMAT. EQUIVANCED TO WTLF AND INDEXED THE SAME AS ARRAY CE. EACH FACTOR = 100000 * COMPONENT WEIGHT / TOTAL COUPLED ITEM WEIGHT.
WMAX	ASINER	WEIGHT OF THAT KIND OF BULK CARGO WHICH REMAINS UNASSIGNED IN GREATEST QUANTITY IN EITHER DIRECTION.
WORD1	LEGPRO	PACKED WORD 1 FOR SHIPPING LEVEL I CARGO ON LOWER LEGS
WORD2	LEGPRO	PACKED WORD 2 FOR SHIPPING LEVEL I CARGO ON LOWER LEGS
WORD3	LEGPRO	PACKED WORD 3 FOR SHIPPING LEVEL I CARGO ON LOWER LEGS
WP	PROPCL	PROPELLANT REQUIRED FOR THIS STAGE TO ACHIEVE REQUIRED VELOCITY VRQ
WPEXP	PERLNK	MAXIMUM PAYLOAD UP IF VEHICLE IS EXPENDED
WPEXP	RDVEH	MAXIMUM PAYLOAD UP ON THIS LEG IF VEHICLE IS EXPENDED
WPLDN	PERLNK	MAXIMUM PAYLOAD DOWN WHEN PAYLOAD UP IS ZERO.
WPLDN	PROPCL	WEIGHT OF PAYLOAD DOWN
WPLDN	RDVEH	MAXIMUM PAYLOAD DOWN ON THIS LEG
WPLUP	PERLNK	MAXIMUM PAYLOAD UP WHEN PAYLOAD DOWN IS ZERO.
WPLUP	PROPCL	PAYLOAD UP
WPLUP	RDVEH	MAXIMUM PAYLOAD UP ON THIS LEG IF VEHICLE IS NOT EXPENDED

WPMAX	PERLNK	TOTAL PROPELLANT WEIGHT FOR WHOLE VEHICLE (SUM OF PROPELLANT WEIGHTS FOR EACH STAGE)
~~WPMX~~	~~PROPCL~~	~~WPMX(I) = MAXIMUM USABLE PROPELLANT WEIGHT FOR STAGE I.~~
WPREQ	PERLNK	WEIGHT OF PROPELLANT REQUIRED
WPREQ	PROPCL	WEIGHT OF REQUIRED PROPELLANT
~~WPREQL~~	~~PERLNK~~	~~VALUE OF WPREQ ON PREVIOUS ITERATION~~
WS	ASINER	WORKING STORAGE MATRIX CONTAINING 2 WORDS FOR EACH CARGO ITEM TO BE ASSIGNED IN EACH DIRECTION
WS	FIND	WORKING STORAGE MATRIX CONTAINING 2 WORDS FOR EACH CARGO ITEM TO BE ASSIGNED IN EACH DIRECTION
WS	LEGPRO	NOT DIRECTLY USED BUT EQUIVALENCED TO FLTA, WHICH IS USED
~~WT~~	~~ASINER~~	~~ITEM WEIGHT (LBS.)~~
WT	FIND	ITEM WEIGHT (LBS.)
WT	LEGPRO	WEIGHT OF HEAVIEST ITEM LEFT IN CAPTURE BIN
WTASK	LEGPRO	ORIGINAL WEIGHT OF BULK CARGO ITEM AS SPECIFIED BY MISSION INPUT
WTDN	LEGPRO	CARGO DOWN WEIGHT
WTDN	RDMISS	CARGO DOWN WEIGHT
~~WTLF~~	~~RDMISS~~	~~UP OR DOWN WEIGHT OF COUPLED COMPONENT TO BE USED IN COMPUTING WEIGHT LOAD~~ FACTORS. FLOATING POINT FORMAT. WTLF IS AN ARRAY EQUIVALENCED TO WLF.
WTLIM	FIND	WEIGHT LIMIT OF ANY ITEM TO BE ASSIGNED
WTMAX	FIND	WEIGHT OF THE LARGEST UNASSIGNED CARGO ITEM FOUND SO FAR WHICH SATISFIES MODE AND OTHER SPECS.
WTMAX	LEGPRO	2-WORD ARRAY CONTAINING MAXIMUM PAYLOAD UP AND DOWN FOR THIS VEHICLE
~~WTOOK~~	~~LEGPRO~~	~~WEIGHT OF PORTION OF BULK CARGO ITEM SUBDIVIDED DURING ASSIGNMENT~~
WTUP	LEGPRO	CARGO UP WEIGHT
WTUP	RDMISS	CARGO UP WEIGHT
~~X~~	~~ASINER~~	~~INDEX VARIABLE~~
X	FIND	SCRATCH VARIABLE
X	LEGPRO	HOLDS CONTAINER RETURN OPTION IN FLOATING POINT
~~X~~	~~PERLNK~~	~~USED TO DETERMINE CONVERGENCE OF ITERATION~~
XFLT	LEGPRO	NUMBER OF FLIGHTS ON THIS LEG/VEHICLE/YEAR (SAME AS NFLT, BUT IN FLOATING POINT FORMAT)
~~XLF~~	~~SPRINT~~	~~MATRIX TO ACCUMULATE FLIGHTS AND VOLUME/LOAD FACTORS FOR EACH VEHICLE/LEG~~
XL1	/ASDAT/	FIRST PART OF CURREN LEG NAME
XL2	/ASDAT/	SECOND PART OF CURRENT LEG NAME
~~XX~~	~~RDSPD~~	~~TEMPORARY VARIABLE~~
Y	ASINER	SCRATCH VARIABLE
Y	FIND	SCRATCH VARIABLE
~~Y~~	~~SPDAP~~	~~REAL VALUE OF L~~
YEAR	LEGPRO	DATE CARGO ITEM IS TO BE SHIPPED
YEAR	READER	FIRST OR LAST YEAR (ABSOLUTE) FOR PREFERENCE VEHICLE
~~YES~~	~~ASINER~~	~~CONTAINS CODED WORD -YES- FOR COMPARISON WITH EXP~~
YRS	RDFAC	MAX YEARS FACILITY LIFE BEFORE REPLACEMENT

Z	SORT	INDEX VARIABLE
ZERO	RDLEG	CONSTANT SET TO 0.

DORCA NOMENCLATURE LIST

COMMON OR ROUTINE NAME	VARIABLE NAME	DESCRIPTION
ASINER	BLKLIM	THIS VARIABLE WHEN APPLIED TO A BULK CONTAINER CAPACITY INDICATES THE MIN. LOAD (LBS.) WHICH MAY BE SHIPPED IN A BULK CONTAINER.
ASINER	CUTOFF	IF VCAP < CUTOFF, VEHICLE IS CONSIDERED FILLED. IF CCAP < CUTOFF, CURRENT CONTAINER IS CONSIDERED FULL AND A NEW CONTAINER IS ASSIGNED.
ASINER	C1	FIRST PART OF CARGO ITEM NAME
ASINER	C2	SECOND PART OF CARGO ITEM NAME
ASINER	DNMAX	MAX. WEIGHT WHICH VEHICLE CAN CARRY DOWNWARD IF IT FLIES UPWARD COMPLETELY EMPTY.
ASINER	EXPMAX	MAX. WEIGHT WHICH VEHICLE CAN CARRY IN EXPENDED MODE
ASINER	I	INDEX VARIABLE
ASINER	ICF	BEGINNING OF CAPTURE BIN IN CARGO TABLE
ASINER	ICL	END OF CAPTURE BIN IN CARGO TABLE
ASINER	ID	DIRECTION OF CURRENT CARGO ITEM
ASINER	IF	INDEX IN CARGO TABLE OF FIRST CARGO ITEM FOR THIS VEH/LEG/YR
ASINER	IL	INDEX IN CARGO TABLE OF LAST CARGO ITEM FOR THIS VEH/LEG/YR
ASINER	IM	MODE OF ITEM CURRENTLY BEING PROCESSED.
ASINER	IP	SCRATCH VARIABLE
ASINER	IRT	1-BIT ROUND-TRIP FLAG IN CARGO TABLE IF IRT = 1, ITEM MUST MAKE ROUND TRIP. IF IRT = 0, ITEM TRAVELS IN ONE DIRECTION ONLY
ASINER	IS	SUBSCRIPT OF CURRENT CARGO ITEM IN DDB ARRAY
ASINER	ISD	ISD=1 IF CARGO ITEM REQUIRES SINGLE DEPLOYMENT, =0 IF MULTIPLE DEPLOYMENT IS A ACCEPTABLE.
ASINER	IT	TYPE OF CURRENT ITEM. SEE ITEM
ASINER	IX	TEMPORARY VARIABLE
ASINER	J	SCRATCH VARIABLE
ASINER	JL	SCRATCH VARIABLE - UPPER BOUND OF DDB CONTAINING DATA FOR CURRENT VEH.
ASINER	JRT	JRT=1 IF CARGO ITEM MUST MAKE ROUND TRIP ON SAME VEHICLE (SAME FLIGHT NUMBER) =0 IF ROUND TRIP MAY BE ASSIGNED TO DIFFERENT FLIGHTS.
ASINER	J1	SCRATCH VARIABLE - LOWER BOUND OF DDB CONTAINING DATA FOR CURRENT VEH.
ASINER	K	INDEX VARIABLE
ASINER	KVEH	NO. OF CURRENT VEHICLE
ASINER	L	SCRATCH VARIABLE
ASINER	LEG	NO. OF CURRENT LEG
ASINER	LIMLEG	LIMIT ON NUMBER OF CARGO ITEMS WHICH MAY BE ASSIGNED TO ANY FLIGHT OF ANY VEHICLE ON THIS LEG.
ASINER	LIMVEH	LIMIT ON NUMBER OF CARGO ITEMS WHICH MAY BE ASSIGNED TO ANY FLIGHT OF THIS VEHICLE.

ASINER	MAXF	MAX. NO. OF FLIGHTS WHICH CAN BE TOTALED IN TW MATRIX
ASINER	MAXI	MAX. NO. OF CARGO ITEMS WHICH CAN BE STORED IN WS MATRIX
ASINER	MBASE	MBASE = 2 IF ASINER HAS ASSIGNED ALL REGULAR CARGO AND IS NOW FILLING UP THE LAST FLIGHT WITH CARGO FROM THE CARGO BIN, MBASE = 3 IF SOME REGULAR CARGO STILL REMAINS UNASSIGNED.
ASINER	N	SCRATCH VARIABLE IDICATING NO. OF CELLS IN DDB USED FOR DATA FOR CURRENT VEHICLE
ASINER	NEV	NUMBER OF EXPENDED VEHICLES REQUESTED FOR THIS LEG/YEAR.
ASINER	NO	CONTAINS THE CODED WORD --NO-- FOR COMPARISON
ASINER	TOTAL	TOTAL AMOUNT OF WHICHEVER KIND OF CARGO REMAINS IN SMALLEST SUPPLY IN SAME DIRECTION AS MANNED CAPSULE TO BE FILLED. USED TO DETERMINE WHICH KIND OF BULK TO LOAD.
ASINER	TVCAP	SCRATCH VARIABLE FOR REMAINING VEHICLE CAPACITY
ASINER	UPMAX	MAX. WEIGHT WHICH VEHICLE CAN CARRY UPWARDS, ASSUMING VEHICLE RETURNS EMPTY.
ASINER	VOLUME	VOLUME FACTOR OF CURRENT CARGO ITEM
ASINER	W	ITEM WEIGHT
ASINER	WMAX	WEIGHT OF THAT KIND OF BULK CARGO WHICH REMAINS UNASSIGNED IN GREATEST QUANTITY IN EITHER DIRECTION.
ASINER	WS	WORKING STORAGE MATRIX CONTAINING 2 WORDS FOR EACH CARGO ITEM TO BE ASSIGNED IN EACH DIRECTION
ASINER	WT	ITEM WEIGHT (LBS.)
ASINER	X	INDEX VARIABLE
ASINER	Y	SCRATCH VARIABLE
ASINER	YES	CONTAINS CODED WORD -YES- FOR COMPARISON WITH EXP
CNTRPT	I	CARGO ELEMENT INDEX OF CURRENT CONTAINER
CNTRPT	ICE	CARGO ELEMENT INDEX OF ITEM IN CARGO TABLE
CNTRPT	ICNT	CONTAINER INDEX OF ITEM IN CARGO TABLE
CNTRPT	IDIR	DIRECTION INDICATOR (0 - UP, 1 - DOWN)
CNTRPT	IFAWD	PACKED WORD TO CREATE ENTRY TO IFA TABLE FOR CONTAINER BEING COSTED
CNTRPT	ILEG	LEG NO. OF CURRENT LEG
CNTRPT	IPASS	FLAG TO DETECT IF ROUTINE IS IN 2ND PASS
CNTRPT	IYR	CURRENT YEAR OF FLIGHT, NORMALIZED TO RLDATE.
CNTRPT	JLEG	LEG NO. OF LEG PRIOR TO CURRENT LEG
CNTRPT	L	LOCATION OF CARGO ITEM IN LEVEL 2 CARGO TABLE
CNTRPT	LINE	ARRAY OF ANNUAL COUNTS FOR CURRENT CONTAINER AND LEG
CNTRPT	LL	LOCATION OF DATA FOR THIS CONTAINER IN CARGO ELEMENT TABLE
CNTRPT	M	RELATIVE YEAR + 3. INDEX TO LINE ARRAY.
CNTRPT	N	RELATIVE YEAR + 3. INDEX TO LINE ARRAY.
CNTRPT	NDEX	IF NONZERO, THIS CONTAINER WAS ALSO INPUT TO FACILITIES TABLE AND IS TO BE ADDED TO IFA TABLE FOR COSTING.
CNTRPT	N1	REPRENTS BOTH YEAR AND COUNT FOR CONTAINER ENTRY TO IFA TABLE
CNTRPT	TOT	TOTAL NO. OF CONTAINERS LEAVING EARTH PER YEAR

CSTRPT	COST1	COST ARRAY. FORMAT- COST NAME (2 WDS), TOTAL FOR ALL YRS (1 WD), TOTAL COST FOR EACH YEAR, 1970-1999 (30 WDS).
CSTRPT	COST2	COST ARRAY SIMILAR TO COST1.
CSTRPT	COST3	COST ARRAY SIMILAR TO COST1.
CSTRPT	COST4	COST ARRAY SIMILAR TO COST1.
CSTRPT	COST5	COST ARRAY SIMILAR TO COST1. NOT USED.
CSTRPT	COST6	COST ARRAY SIMILAR TO COST1.
CSTRPT	FLTAC	A MATRIX OF VEHICLE ACQUISITIONS BY VEHICLE TYPE AND BY YEAR
CSTRPT	I	INDEX VARIABLE
CSTRPT	ICE	LOC. OF CARGO ELEMENT IN DDB TABLE
CSTRPT	IFAC	ARRAY INDICATING PROGRAM/MISSION WHICH FIRST USED EACH FACILITY.
CSTRPT	IFAT	INDEX OF THIS FACILITY.
CSTRPT	IFLG	FLAG TO ROUTE LOGIC FLOW (3 - NEW PROGRAM, 2 - SAME PROGRAM BUT NEW MISSION, 1 - SAME PROGRAM AND MISSION BUT NEW VEHICLE)
CSTRPT	II	TEMPORARY VARIABLE
CSTRPT	IL	FLAG INDICATING WHETHER ANOTHER ENTRY SHOULD BE MADE IN IFAC TABLE.
CSTRPT	IMIS	INDEX TO THE MISSION NAME IN THE MISSION TABLE
CSTRPT	IPM	PACKED WORD GIVING CURRENT PROGRAM AND MISSION
CSTRPT	IPRNT	ONE OF THE FLAGS WHICH CONTROL WHEN COSTS ARE PRINTED
CSTRPT	IPRO	INDEX TO THE PROGRAM NAME IN THE PROGRAM TABLE
CSTRPT	IVEH	POINTER TO BEGINNING OF DATA FOR CURRENT VEHICLE IN DDB. ALSO USED AS A VEHICLE NUMBER.
CSTRPT	J	INDEX VARIABLE
CSTRPT	JF	POINTER TO CURRENT POSITION IN FACILITY ACQUISITION TABLE (IFA).
CSTRPT	KFLAG	SAME AS IFLAG(11). IF KFLAG=1, REPORT WILL BE PRINTED IN 8X11-INCH FORMAT, OTHERWISE IN 8X14-INCH FORMAT.
CSTRPT	L	INDEX VARIABLE
CSTRPT	LL	INDEX VARIABLE
CSTRPT	LX	LOOP VARIABLE (1 - FACILITY COSTS, 2 - VEHICLE OPERATIONS COSTS)
CSTRPT	L1	LOCATION OF SPREAD FUNCTION FOR VEHICLE/FACILITY DEVELOPMENT COSTS
CSTRPT	L2	LOCATION OF SPREAD FUNCTION FOR VEHICLE/FACILITY PRODUCTION COST
CSTRPT	M	LOC. OF FACILITY INFO TO BE CURRENTLY USED
CSTRPT	N	NO. OF FACILITY BEING CONSIDERED
CSTRPT	NP	POINTER TO BEGINNING OF CARGO BLOCK IN DDB FOR THIS PROGRAM, MISSION AND VEHICLE.
CSTRPT	NCEP	CARGO ELEMET NO.
CSTRPT	NIF	NUMBER OF FACILITIES ENTERED IN IFAC ARRAY.
CSTRPT	NL	POINTER TO END OF CARGO BLOCK IN DDB FOR THIS PROGRAM, MISSION, AND VEHICLE.
CSTRPT	NM	MISSION NO.
CSTRPT	NPM	PROGRAM/MISSION CORRESPONDING TO DATA STORED IN ARRAY LINE.
CSTRPT	NTW	NUMBER OF YEARS IN THE TIME PERIOD WHICH WILL BE PRINTED
CSTRPT	NV	POINTER TO BEGINNING OF DATA FOR THIS VEHICLE IN DDB.

CSTRPT	NYR	YEAR IN WHICH FACILITY IS USED
CSTRPT	VDATA	ARRAY CONTAINING NAME (2 WDS), TOTAL COST OR COUNT FOR ALL YRS (1 WD), AND COST OR COUNT FOR EACH YEAR, 1970-1999 (30 WDS.) EQUIVALENCED TO ARRAY LINE.
DDBSFT	MOSC	NUMBER OF EMPTY CELLS IN DDB BETWEEN END OF CARGO ELEMENT DATA AND BEGINNING OF LEVEL I CARGO TABLE.
DDBSFT	MOSF	NUMBER OF EMPTY CELLS IN DDB BETWEEN END OF FACILITY TABLE AND BEGINNING OF CARGO ELEMENT TABLE
DDBSFT	NSFT	NUMBER OF CELLS TO SHIFT CARGO TABLE.
DORCA	CE	CORE EXTENSION TO DYNAMIC DATA BLOCK (SEE DDB DEFINITION). ARRAY CE IS DIMENSIONED AND DEFINED IN BLANK COMMON ONLY IN MAIN ROUTINE DORCA. IT ADDS 35000 CELLS ONTO DDB. PURPOSE OF CE IS TO FACILITATE CHANGING THE SIZE OF DDB BY ALTERING ONLY THE DIMENSION OF CE AND THE VALUE OF LOWCOR IN DORCA ONLY.
DORCA	I	INDEX VARIABLE
DORCA	IGAP	NUMBER OF UNUSED CELLS IN DDB ARRAY
DORCA	JCASE	COUNT OF CASES DORCA HAS ATTEMPTED TO EXECUTE
DPAGER	I	INDEX VARIABLE
DPAGER	LINE	ARRAY CONTAINING COST DATA FOR EACH YEAR
FACNUM	A	ARRAY CONTAINING FACILITY NAME (2 WDS), TOTAL ACQUIRED FOR ALL YEARS (1 WD), AND NUMBER ACQUIRED IN EACH YEAR 1970-1999 (30 WDS)
FACNUM	I	INDEX VARIABLE
FACNUM	ICE	LOCATION OF DATA IN DDB FOR CURRENT CARGO ELEMENT
FACNUM	IFAC	NO. OF FACILITY BEING CONSIDERED
FACNUM	JF	POINTER TO CURRENT POSITION IN FACILITY ACQUISITION TABLE IFA
FACNUM	L	INDEX VARIABLE = NYR-FYEAR+4
FACNUM	M	LOC. OF FACILITY INFO IN THE DDB ARRAY BEING CONSIDERED
FACNUM	N	CARGO ELEMENT NO. BEING CONSIDERED
FACNUM	NCNT	NO. OF FACILITIES SHIPPED THAT YEAR
FACNUM	NPMCE	PACKED PROGRAM, MISSION AND CARGO ELEMENT NO.S BEING CONSIDERED
FACNUM	NPMCEL	PACKED PROG., MISSION AND CARGO ELE. NO.S PREVIOUSLY CONSIDERED
FACNUM	NYR	YEAR IN WHICH THE PROG./MISS/CE TAKES PLACE
FACRPT	A	AN ARRAY OF THE TOTAL NO. FACILIES PER YEAR EQUIVALENT TO LINE
FACRPT	I	INDEX VARIABLE
FACRPT	JF	INTEGER POINTER TO FACILITY INFORMATION TO BE CONSIDERED CURRENTLY
FACRPT	NCEL	CARGO ELEMENT NO. OF LAST FACILITY CONSIDERED BEFORE NCE
FACRPT	NCEP	CARGO ELEMET NO. OF THE PRESENTLY CONSIDERED FACILITY
FACRPT	NM	MISSION NO. OF FACILITY CURRENTLY BEING CONSIDERED
FACRPT	NML	MISSION NO. OF LAST FACILITY CONSIDERED BEFORE NM
FACRPT	NP	PROG. NO. OF FACILITY CURRENTLY BEING CONSIDERED
FACRPT	NPL	PROG. NO. OF LAST FACILITY CONSIDERED BEFORE NP
FIND	CONTRE	IF CONTRE = 0, BULK CONTAINERS ARE RETURNED (EMPTY)

		IF CONTRE ≠ 0, CONTAINERS ARE EXPENDED
FIND	CW	CONTAINER WEIGHT
FIND	FLAG	IF NONZERO, INDICATES CONTAINER ASSIGNED AND SHOULD BE ENTERED INTO THE CR TABLE AND ACCOUNTED IN RUNNING TOTALS
FIND	I	INDEX VARIABLE
FIND	ID	DIRECTION OF CURRENT CARGO ITEM
FIND	IS	SUBSCRIPT OF CURRENT CARGO ITEM IN DDB ARRAY
FIND	ISD	INDICATES WHETHER CARGO ITEM REQUIRES SINGLE DEPLOYMENT (ISD=1) OR MULTIPLE DEPLOYMENT (ISD=0)
FIND	IX	TEMPORARY VARIABLE
FIND	JRT	IF JRT=1, CARGO MUST MAKE ROUND TRIP ON SAME FLIGHT
FIND	K	INDEX VARIABLE
FIND	LIMCAL	MAXIMUM NUMBER OF CONSECUTIVE UNSUCCESSFUL CALLS TO SUBROUTINE -FIND- BEFORE PROGRAM ABORTS ITSELF. LIMIT IS 100.
FIND	MAXA	MAX. ASSIGN. WHICH CAN BE LISTED IN FLTA MATRIX FOR LEG/VEH/YEAR
FIND	MAXC	MAX ALLOWABLE CONTIANERS FOR LEG/VEH/YEAR
FIND	MTD	PACKED WORD CONTAINING MODE, TYPE AND DIRECTION OF DESIRED CARGO
FIND	NB	SCRATCH VARIABLE INDICATING BIT POSITION FOR PACKING AND UNPACKING DATA IN CR ARRAY.
FIND	TCAP	TEMPORARY VARAIBLE CONTAINING UNUSED CONTAINER CAPACITY
FIND	VOLUME	VOLUME FACTOR OF CURRENT CARGO ITEM
FIND	W	ITEM WEIGHT
FIND	WEIGHT	EQUIVALENT UP-WEIGHT OF CARGO PLUS ITS CONTAINER (IF ANY) ON ROUND TRIP.
FIND	WS	WORKING STORAGE MATRIX CONTAINING 2 WORDS FOR EACH CARGO ITEM TO BE ASSIGNED IN EACH DIRECTION
FIND	WT	ITEM WEIGHT (LBS.)
FIND	WTLIM	WEIGHT LIMIT OF ANY ITEM TO BE ASSIGNED
FIND	WTMAX	WEIGHT OF THE LARGEST UNASSIGNED CARGO ITEM FOUND SO FAR WHICH SATISFIES MODE AND OTHER SPECS.
FIND	X	SCRATCH VARIABLE
FIND	Y	SCRATCH VARIABLE
FINDSP	I	INDEX VARIABLE
FINDSP	IPT	LOCATION OF START OF DATA FOR THIS SPREAD FUNCTION IN DDB ARRAY.
FINDSP	NAME	NAME OF SPREAD FUNCTION (A6, A4 FORMAT)
FINDSP	ONE	DEFAULT VALUE FOR SPREAD TABLE POINTER IPT
INPRO	CASUC	CONTENTS OF COLUMNS 11-20 OF INPUT CARD, USUALLY A TABLE NAME.
INPRO	INIT	FLAG INDICATING FIRST TIME THROUGH
INPRO	NAME	CONTENTS OF COLUMNS 1-10 OF INPUT CARD, USUALLY THE WORD TABLE.
LEGPRO	A	WEIGHT LOAD FACTOR (FLTG PT)
LEGPRO	BCELF	BULK CARGO LOAD FACTOR
LEGPRO	BOX	CONTAINS CODED WORD -GB BOX- FOR NAME OF NEWLY CREATED CARGO ELEMENTS
LEGPRO	BOX2	CONTAINS CODED WORD -NL BOX- FOR NAME OF NEWLY CREATED CARGO ELEMENTS
LEGPRO	CONT	PACKED FIRST WORD FOR CONTAINER BEING ADDED TO PHASE I CARGO TABLE

LEGPRO	CONT2	PACKED SECOND WORD FOR CONTAINER BEING ADDED TO PHASE I CARGO TABLE
LEGPRO	DISCRP	7-WORD ARRAY OF CODED DESCRIPTION FOR NEWLY CREATED CARGO ELEMENTS
LEGPRO	DNMAX	MAXIMUM PAYLOAD DOWN FOR THIS VEHICLE
LEGPRO	DW	TINY WEIGHT INCREMENT SUCH THAT, WHEN LEGPRO IS SEARCHING THE CAPTURE BIN FOR THE HEAVIEST ITEM, ITEMS OF EQUAL WEIGHT ARE ASSIGNED PRIORITIES AS FOLLOWS- ROUND TRIP SAME VEHICLE FLIGHT (HIGHEST PRIORITY) ROUND TRIP ON DIFFERENT VEHICLES DOWN ONLY UP ONLY (LOWEST PRIORITY)
LEGPRO	EXPMAX	MAXIMUM PAYLOAD UP FOR THIS VEHICLE IF EXPENDED
LEGPRO	FLAGS	SIX ONE-BIT FLAGS PACKED INTO BOTH LEVEL I AND LEVEL II CARGO TABLES
LEGPRO	FLTNO	CURRENT FLIGHT NO.
LEGPRO	GBVOL	PAYLOAD VOLUMES UP AND DOWN (2 WDS)
LEGPRO	GBWT	PAYLOAD WEIGHTS (PRIMARY ITEMS ONLY) UP AND DOWN (2 WDS)
LEGPRO	GBWTV	2-WORD LIST OF VEHICLE WEIGHTS UP AND DOWN (GROUND BASE MODE)
LEGPRO	I	INDEX VARIABLE
LEGPRO	IBOX	USED TO CREATE UNIQUE SYMBOL FOR NEW COUPLED CARGO BOX NAME
LEGPRO	IBVEH	BEGINNING OF DATA CURRENTLY BEING CONSIDERED
LEGPRO	ICAT	CATEGORY OF CARGO ELEMENT
LEGPRO	ICB	ICB=1 IF THERE IS A CAPTURE BIN FOR THIS LEG/YR, =0 IF NOT
LEGPRO	ICC	BITS 0-1 OF WORD 3 OF LEVEL I CARGO ITEM (00-REGULAR NON-COUPLED ITEM, 01-IMPOSSIBLE, 10-WHOLE COUPLED COMPOSITE, 11-COMPONENT OF COUPLED ITEM)
LEGPRO	ICE	INDEX OF ELEMENT IN CARGO ELEMENT TABLE
LEGPRO	ICF	BEGINNING OF CAPTURE BIN IN LEVEL I CARGO TABLE
LEGPRO	ICGN	COMPOSITE GROUP NUMBER
LEGPRO	ICI	LOOP COUNTER FOR LOOP THROUGH A PRIORI CARGO LIST
LEGPRO	ICL	END OF CAPTURE BIN IN CARGO TABLE
LEGPRO	ICLS	CONTAINER CLASS OF CARGO ITEM
LEGPRO	ICPL	FLAG SHOWING WHETHER LEGPRO HAS (ICPL=1) OR HAS NOT (ICPL) FOUND THE COMPONENTS OF A COUPLED ITEM (THOSE SECONDARIES WITH SAME C.G.N. AS THE PRIMARY)
LEGPRO	IC2	BEGINNING OF COUPLED ITEMS (SECONDARIES) FOR THIS LEG/YR. NOT TO BE SHIPPED BY ASINER.
LEGPRO	IC3	END OF COUPLED ITEMS (SECONDARIES) FOR THIS LEG/YR
LEGPRO	IDATE	AN INTEGER VALUE OF RLDATE
LEGPRO	IDIR	DIRECTION (0-UP, 1-DOWN -OR- 1-UP, 2-DOWN)
LEGPRO	IDV	VELOCITY INCREMENT NECESSARY FOR VEHICLES ON THIS LEG
LEGPRO	IF	LOCATION OF FIRST CARGO ITEM IN DDB FOR THIS LEG/VEHICLE/YEAR
LEGPRO	IFD	FLIGHT NO. DOWN OF CONTAINER
LEGPRO	IFLG	FLAG USED TO PACK DATA INDICATING DIRECTION OF CARGO ITEM
LEGPRO	IFLT	EQUAL TO EITHER IFD OR IFU
LEGPRO	IFU	FLIGHT NO. UP OF CONTAINER
LEGPRO	IFY	FIRST YEAR FOR VEHICLE PREFERENCE
LEGPRO	IGAP	NUMBER OF CORE CELLS FREED BY DELETION FROM CARGO TABLE OF ITEMS (REGULAR AND

```
                    CAPTURE) JUST SCHEDULED BY ASDNER
LEGPRO   II         POINTER TO HEAVIEST ITEM LEFT IN CAPTURE BIN
LEGPRO   IL         LOCATION OF LAST CARGO ITEM IN DDB FOR THIS LEG/VEHICLE/YEAR
LEGPRO   ILL        SAVED POINTER TO POSITION IN CARGO TABLE
LEGPRO   ILY        LAST YEAR FOR PREFERENCE OF THIS YEAR
LEGPRO   IM         RUNNING VARIABLE FOR LOOP ON CAPTURE BIN
LEGPRO   IMAX       RUNNING INDEX THROUGH CAPTURE BIN
LEGPRO   IPREF      RUNNING INDEX THROUGH VEHICLE PREFERENCE LIST
LEGPRO   IROUND     INTEGER 100, USED TO ROUND PROPELLANT OFF-LOADED UPWARDS TO NEAREST 100 LBS
LEGPRO   IRTDN      COMPOSITE FLAG CONTAINING ROUND-TRIP AND DIRECTION FLAGS OF CURRENT CARGO ITEM
                    (1 BIT EACH). 2-ROUND TRIP, 1-DOWN ONLY, 0-UP ONLY
LEGPRO   IS         LOCATION OF THIS CARGO AT ITS SOURCE IN PHASE I CARGO TABLE
LEGPRO   ISAVE      TEMPORARY STORAGE FOR CONTENTS OF IFLAG(4), WHICH IS ALTERED
LEGPRO   ISKIP      ISKIP = 0 FOR VEHICLE UPPER STAGE IN UP DIRECTION
                            1 FOR COUPLED ITEM UP
                            2 VEHICLE UPPER STAGE GOING DOWN THE LEG
                            3 COUPLED ITEM GOING DOWN
                            4-8 FOR LOWER VEHICLE STAGES
LEGPRO   IVEH       INDEX OF VEHICLE WET STAGE
LEGPRO   IVH        VEHICLE INDEX EXTRACTED FROM LEVEL I CARGO ITEM
LEGPRO   I1         LOOP VARIABLE
LEGPRO   I2         BIT POSITION FOR UNPACKING STAGE NUMBERS
LEGPRO   JBOX       USED TO CREATE UNIQUE SYMBOL FOR NAME OF NEW COUPLED CARGO BOX
LEGPRO   JCGN       2-WORD LIST OF COMPOSITE GROUP NUMBER UP AND DOWN
LEGPRO   JCI        CONTAINS BITS 0 AND 1 TO BE PACKED INTO WORD 3 OF LEVEL II DATA. 2-WORD ARRAY
                    FOR UP AND DOWN TRIPS
LEGPRO   JFLT       FLIGHT NUMBER TO INSERT INTO LEVEL II CARGO TABLE.
LEGPRO   JL         END OF DATA FOR THIS VEHICLE IN DDB
LEGPRO   JN         NUMBER OF WORDS OF DATA FOR THIS VEHICLE IN DDB
LEGPRO   JVEH       VEHICLE SELECTED FROM PREFERENCE LIST FOR CAPTURE CARGO IF AN EXPENDED FLIGHT
                    IS NECESSARY.
LEGPRO   JWORD      PACKED WORD IN FORMAT OF THIRD WORD OF LEVEL I CARGO ITEM (COUPLED)
LEGPRO   J1         START OF DATA FOR THIS VEHICLE IN DDB
LEGPRO   KBOX       KLUDGE FACTOR TO CREATE COUPLED CARGO BOX NAME
LEGPRO   KVEH       VEHICLE SELECTED WHEN ONLY A CAPTURE BIN IS LEFT FOR THIS LEG/YEAR (NO CARGO
                    WITH A SPECIFIED VEHICLE)
LEGPRO   LASTCE     INDEX OF LAST CARGO ELEMENT INPUT (NOT CREATED BY LEGPRO)
LEGPRO   LEGNO      LEG NUMBER OF NEXT BLOCK OF CARGO ITEMS.
LEGPRO   LEG1       LEG NUMBER OF CURRENT BLOCK OF CARGO ITEMS.
LEGPRO   LF         100,000 TIMES THE LOAD FACTOR
LEGPRO   LFB        100,000 TIMES THE BULK LOAD FACTOR
LEGPRO   LFGB       LOAD FACTOR * 100000 FOR GROUND BASE OPERATIONS
LEGPRO   LGYR1      LEG/YEAR OF A CARGO ITEM
```

LEGPRO	LGYR1	LEG/YEAR OF FIRST CARGO ITEM
LEGPRO	LIML2	MAX NUMBER OF LEVEL II ITEMS WHICH CAN FIT IN 510-WORD BUFFER
LEGPRO	LOADFC	LOAD FACTOR
LEGPRO	LS	LONGSHORING FLAG
LEGPRO	LVEH	DEFAULT VEHICLE FOR NEXT LOWER LEG
LEGPRO	LX	LOCATION OF CARGO WEIGHT FOR CURRENT DIRECTION
LEGPRO	L1	LEG NAME, FIRST PART
LEGPRO	L2	LEG NAME, LAST PART
LEGPRO	L3	NAME OF NEXT LOWER LEG (FIRST 6 CHARACTERS)
LEGPRO	L4	NAME OF NEXT LOWER LEG (LAST 4 CHARACTERS)
LEGPRO	M	INDEX VARIABLE
LEGPRO	NBASE	LOCATION OF START OF DATA FOR THIS CARGO ELEMENT
LEGPRO	NCGN	COMPOSITE GROUP NUMBER
LEGPRO	NDEX	CONTAINER NO. THAT IS BEING SHIPPED
LEGPRO	NDEX1	CURRENT INDEX OF LEG/VEH/YEAR
LEGPRO	NDEX2	NEXT INDEX OF LEG/VEH/YEAR
LEGPRO	NDEX3	LAST LEG/YR
LEGPRO	NDEX4	CURRENT LEG/YR
LEGPRO	NDEX5	NEXT LEG/YR
LEGPRO	NDXCE	INDEX OF CARGO ELEMENT TABLE
LEGPRO	NEWLEG	INDEX OF LOWER LEG FOR THIS CARGO GROUP
LEGPRO	NF	EQUALS NIFA
LEGPRO	NOBOX	USED TO CREATE UNIQUE SYMBOL FOR NAME OF NEW COUPLED CARGO BOX
LEGPRO	NOFLT	TOTAL NUMBER OF FLIGHTS OF VEHICLE = VEHNDX ON CURRENT LEG, INCLUDING ANY RESULTING FROM PREVIOUS CALLS TO ASINER
LEGPRO	NPF	NUMBER OF PROPELLANT TANKS NEEDED TO FUEL FLIGHTS SCHEDULED FOR THIS LEG, VEHICLE AND YEAR.
LEGPRO	NV	EQUALS NIVA
LEGPRO	NWT	WEIGHT LOAD FACTOR FOR VEHICLE, X 100000 (INTEGER)
LEGPRO	NXTLEG	INDEX OF LOWER LEG FOR NEXT CARGO GROUP
LEGPRO	PNMN	PROG/MISSION COMBINATION
LEGPRO	PROP	PACKED FIRST WORD FOR PROPELLANT TANK ADDED TO PHASE I CARGO TABLE
LEGPRO	PROPOL	WEIGHT OF PROPELLANT OFF-LOADED INTO LAST STAGE OF VEHICLE
LEGPRO	PROPWT	WEIGHT OF PROPELLANT NEED TO FUEL ONE FLIGHT OF THIS VEHICLE WHEN FULLY LOADED
LEGPRO	PROP2	PACKED SECOND WORD FOR PROPELLANT TANK ADDED TO PHASE I CARGO TABLE
LEGPRO	RELDT	DATE CARGO ITEM IS TO BE SHIPPED
LEGPRO	TEMP	TEMPORARY VARIABLE USED IN DATA INTERCHANGE
LEGPRO	TIME	= 0, IF INTIAL RUN-THROUGH FOR VEH/LEG/YEAR
LEGPRO	TOTWT	TOTAL WEIGHT CARRIED BY VEHICLE UP AND DOWN ON CURRENT FLIGHT, EXPRESSED AS EQUIVALENT UP-WEIGHT.
LEGPRO	TPROPW	ARRAY GIVING TOTAL PROPELLANT REQUIRED FOR ALL FLIGHTS OF EACH VEHICLE ON THIS LEG/YEAR. TPROPW(I) = TOTAL PROPELLANT FOR VEHICLE NO. I

LEGPRO	UPMAX	MAXIMUM PAYLOAD UP FOR THIS VEHICLE
LEGPRO	VEHNDX	CURRENT VEHICLE NO. IN DDR ARRAY
LEGPRO	VEHNO	NEXT VEHICLE NO. IN DDB ARRAY
LEGPRO	VOLV	VEHICLE VOLUME CAPACITY
LEGPRO	VOL1	VOLUME CAPACITY OF CURRENT VEHICLE (VEHNDX)
LEGPRO	VOL2	VOLUME CAPACITY OF VEHICLE NO. LVEH (LOWER LEG DEFAULT VEHICLE)
LEGPRO	VV	PACKED VEHICLE INDICES FROM SECOND AND THIRD LOWER LEG (6 BITS EACH)
LEGPRO	V1	INDEX OF VEHICLE USED ON CURRENT LEG
LEGPRO	V2	INDEX OF VEHICLE USED ON NEXT LOWER LEG
LEGPRO	V3	INDEX OF VEHICLE USED ON SECOND LOWER LEG, IF ANY
LEGPRO	V4	INDEX OF VEHICLE USED ON THIRD LOWER LEG, IF ANY
LEGPRO	WORD1	PACKED WORD 1 FOR SHIPPING LEVEL I CARGO ON LOWER LEGS
LEGPRO	WORD3	PACKED WORD 3 FOR SHIPPING LEVEL I CARGO ON LOWER LEGS
LEGPRO	WS	NOT DIRECTLY USED BUT EQUIVALENCED TO FLTA, WHICH IS USED
LEGPRO	WT	WEIGHT OF HEAVIEST ITEM LEFT IN CAPTURE BIN
LEGPRO	WTASK	ORIGINAL WEIGHT OF BULK CARGO ITEM AS SPECIFIED BY MISSION INPUT
LEGPRO	WTDN	CARGO DOWN WEIGHT
LEGPRO	WTMAX	2-WORD ARRAY CONTAINING MAXIMUM PAYLOAD UP AND DOWN FOR THIS VEHICLE
LEGPRO	WTOOK	WEIGHT OF PORTION OF BULK CARGO ITEM SUBDIVIDED DURING ASSIGNMENT
LEGPRO	WTUP	CARGO UP WEIGHT
LEGPRO	X	HOLDS CONTAINER RETURN OPTION IN FLOATING POINT
LEGPRO	XFLT	NUMBER OF FLIGHTS ON THIS LEG/VEHICLE/YEAR (SAME AS NFLT, BUT IN FLOATING POINT FORMAT)
LEGPRO	YEAR	DATE CARGO ITEM IS TO BE SHIPPED
MERGE	COLS	NUMBER OF COLUMNS IN MATRIX FILE WHICH CONTAIN DATA TO BE MERGED.
MERGE	FILE	MATRIX CONTAINING UP TO 10 LOGICAL RECORDS OF DATA TO BE MERGED.
MERGE	FLOW	INTEGER USED FOR FLOW CONTROL IN ASSIGNED GO TO.
MERGE	I	INDEX VARIABLE.
MERGE	ICOL	POINTER TO COLUMN OF MATRIX FILE CURRENTLY BEING PROCESSED.
MERGE	IMIN	NUMBER OF COLUMN CONTAINING MINIMUM ELEMENT AND INDICATING DATA TO BE ADDED TO MERGE BUFFER MBUFF.
MERGE	INDEX	INDEX NUMBER OF SCRATCH TAPE CONTAINING DATA TO BE MERGED.
MERGE	INTAPE	LOGICAL NUMBER OF TAPE/DISK FILE CONTAINING ORIGINAL INPUT MATRIX.
MERGE	IN1	INDEX NUMBER OF FIRST SCRATCH TAPE CONTAINING SOME DATA.
MERGE	IN2	INDEX NUMBER OF SECOND SCRATCH TAPE CONTAINING SOME DATA.
MERGE	IP	POINTER TO LOCATION OF DATA ELEMENT IN MINIMUM TEST.
MERGE	ITAPE	LOGICAL TAPE NUMBER OF SCRATCH FILE CONTAINING SOME DATA.
MERGE	J	INDEX VARIABLE
MERGE	KEL	COLUMN ELEMENT ON WHICH SORT IS MADE (SAME AS SUBROUTINE ARGUMENT KELT)
MERGE	LNTH	LENGTH OF COLUMN OF MATRIX FILE (510 UNLESS OTHERWISE CHANGED)
MERGE	M	NUMBER OF MATRIX COLUMNS TO READ FROM SCRATCH FILE.
MERGE	MBUFF	ARRAY CONTAINING DATA MERGED FROM COLUMNS OF MATRIX FILE.
MERGE	MINWD	DATA ELEMENT WHICH IS MINIMUM SO FAR.

MERGE	MP	POINTER TO NEXT POSITION IN MERGE BUFFER MBUFF TO BE FILLED.
MERGE	N	NUMBER OF MATRIX COLUMNS TO BE READ FROM INPUT TAPE.
MERGE	NCOLS	NUMBER OF COLUMNS OF INPUT MATRIX STILL REMAINING ON INPUT TAPE.
MERGE	NCPR	NUMBER OF COLUMNS OF INPUT MATRIX PER FULL LOGICAL RECORD ON INPUT TAPE.
MERGE	NREAD	NUMBER OF LOGICAL RECORDS TO READ FROM INPUT FILE INTO FILE MATRIX.
MERGE	NROWS	NUMBER OF ROWS IN INPUT MATRIX. SAME AS SUBROUTINE ARGUMENT NR.
MERGE	NWORDS	NUMBER OF WORDS TO READ FROM INPUT OR SCRATCH TAPE IN CURRENT LOGICAL RECORD
MERGE	NWPR	NUMBER OF WORDS IN A FULL LOGICAL RECORD ON INPUT FILE.
MERGE	OUT	INDEX OF OUTPUT TAPE FOR CURRENT MERGE.
MERGE	OUTAPE	LOGICAL NUMBER OF OUTPUT TAPE FOR CURRENT MERGE.
MERGE	P	P(I,J) CONTAININS POINTERS CONCERNING COLUMN J OF MATRIX FILE.
MERGE	TDAT	TDAT(I,J) GIVES DATA ON FILE INDEX J. J = 1,2,3 ARE SCRATCH TAPES, J = 4 IS THE INPUT TAPE.
PACK	B	THE BIT NO. TO START PACK/UNPACKING
PACK	NBITS	NO. OF BITS HANDLED IN PACK
PACK	PW	THE PACKED WORD
PACK	V	THE VARIABLE FROM/TO WHICH THE BITS ARE UNPACKED/PACKED
PERLNK	ARRAY	ARRAY OF 8 WORDS FOR EACH OF 1 - 6 VEHICLE STAGES: 1 - WSD, 2 - WNUP, 3 - WPMAX, 4 - WINT, 5 - WPBO, 6 - WNIE, 7 - WACP, 8 - ISP ARRAY OF 8 WORDS FOR EACH OF 1 - 6 VEHICLE STAGES: 1 - WSD, 2 - WNUP, 3 - WPMAX, 4 - WINT, 5 - WPBO, 6 - WNIE, 7 - WACP, 8 - ISP
PERLNK	FDV	SAFETY FACTORY FOR ENGINE PERFORMANCE DATA
PERLNK	ISPS	SUM OF ISP NUMBERS FOR ALL STAGES (TOTAL SPECIFIC IMPULSE FOR VEHICLE)
PERLNK	I1	BIT POSITION TO START UNPACKING
PERLNK	J1	LOCATION IN DDB WHERE ISP DATA FOR THIS VEHICLE STAGE SHOULD START
PERLNK	KALL	NUMBER OF TIMES PROPCL WAS CALLED (NUMBER OF ITERATIONS)
PERLNK	M	M = 1 FOR TOTALLY EXPENDABLE VEHICLE, M = 2 FOR EXPENDABLE UPPER STAGE ONLY, M = 3 FOR TOTALLY REUSABLE VEHICLE.
PERLNK	NBVEH	START OF DATA FOR THIS VEHICLE IN DDB
PERLNK	NREQ	NUMBER OF STAGES REQUIRED. SHOULD EQUAL NTOT.
PERLNK	NTOT	NUMBER OF WET STAGES IN THIS VEHICLE
PERLNK	NV	INDEX OF THIS WET STAGE IN VEHICLE TABLE.
PERLNK	PL	ASSUMES VARIOUS VALUES OF PAYLOAD WEIGHT DURING THE ITERATION FOR WPREQ
PERLNK	PLL	VALUE OF PL TWO ITERATIONS AGO
PERLNK	PLMX	TEMPORARY STORAGE FOR WPLUP, WHICH IS CHANGED WHILE WPLDN IS BEING COMPUTED
PERLNK	PLN	VALUE OF PL ON LAST ITERATION
PERLNK	R	IN ROCKET EQUATION, REPRESENTS RATIO OF INITIAL WEIGHT TO END WEIGHT DURING BURNING OF STAGE.
PERLNK	VIN	VELOCITY INCREMENT DELTA V REQUIRED TO TRANSFER FROM ORBIT AT LEG BOTTOM TO ORBIT AT TOP OF LEG (INDEPENDENT OF VEHICLE)
PERLNK	WBO	WEIGHT OF STAGE AT BURNOUT, INCLUDING NON-USABLE FUEL.
PERLNK	WPEXP	MAXIMUM PAYLOAD UP IF VEHICLE IS EXPENDED
PERLNK	WPLDN	MAXIMUM PAYLOAD DOWN WHEN PAYLOAD UP IS ZERO.

PERLNK	WPLUP	MAXIMUM PAYLOAD UP WHEN PAYLOAD DOWN IS ZERO.
PERLNK	WPMAX	TOTAL PROPELLANT WEIGHT FOR WHOLE VEHICLE (SUM OF PROPELLANT WEIGHTS FOR EACH STAGE)
PERLNK	WPREQ	WEIGHT OF PROPELLANT REQUIRED
PERLNK	WPREQL	VALUE OF WPREQ ON PREVIOUS ITERATION
PERLNK	X	USED TO DETERMINE CONVERGENCE OF ITERATION
PROLNK	APRAY	SUM OF ISP NUMBERS FOR ALL STAGES (TOTAL SPECIFIC IMPULSE FOR VEHICLE)
PROLNK	DV	VELOCITY INCREMENT DELTA V REQUIRED FOR ORBIT CHANGE (LEG TRAVEL)
PROLNK	FDV	SAFETY FACTORY FOR ENGINE PERFORMANCE DATA
PROLNK	IVEH	VEHICLE INDEX
PROLNK	I2	BIT POSITION FOR UNPACKING
PROLNK	JVEH	INDEX OF VEHICLE STAGE
PROLNK	JX	BEGINNING OF DATA FOR VEHICLE JVEH IN DDB
PROLNK	JY	BEGINNING OF DATA VEHICLE IVEH IN DDB
PROLNK	J1	LOCATION OF START OF ISP DATA FOR VEHICLE JVEH
PROLNK	M	M = 1 FOR TOTALLY EXPENDABLE VEHICLE, M = 2 FOR EXPENDABLE UPPER STAGE ONLY, M = 3 FOR TOTALLY REUSABLE VEHICLE.
PROLNK	NEXP	EQUALS 1 IF VEHICLE IS TO BE EXPENDED.
PROLNK	NPU	NUMBER OF PROPULSIVE (WET) STAGES
PROLNK	NREQ	NUMBER OF STAGES REQUIRED
PROLNK	PROP	TOTAL PROPELLANT AVAILABLE TO WHOLE VEHICLE (SUM OF MAXIMUM PROPELLANT FOR EACH STAGE)
PROLNK	PROPOL	AMOUNT OF PROPELLANT OFF-LOADED IN LAST STAGE
PROLNK	PROPWT	TOTAL PROPELLANT WEIGHT REQUIRED (COMPUTED BY PROLNK)
PROLNK	TWDN	TOTAL PAYLOAD WEIGHT DOWN
PROLNK	TWUP	TOTAL PAYLOAD WEIGHT UP
PROPCL	ARRAY	ARRAY OF 8 WORDS FOR EACH OF 1 - 6 VEHICLE STAGES: WORD 1 - WSD (DRY STRUCTURE WEIGHT) 2 - WNUP (NON-USABLE PROPELLANT WEIGHT) 3 - WPMAX (MAXIMUM PROPELLANT WEIGHT) 4 - WINT (INTERSTAGE WEIGHT) 5 - WPBO (BOIL-OFF WEIGHT) 6 - WNIE (NON-IMPULSIVE PROPELLANT WEIGHT) 7 - WACP (ATTITUDE CONTROL PROPELLANT WEIGHT) 8 - ISP NUMBER (SPECIFIC IMPULSE)
PROPCL	G	GRAVITY CONSTANT (FT/SEC/SEC)
PROPCL	ISP	ARRAY GIVING SPECIFIC IMPULSE FOR EACH STAGE
PROPCL	ISPEF	ISPEF(I) = EFFECTIVE SPECIFIC IMPULSE FOR STAGE I.
PROPCL	JV	IF JV=1, THIS IS THE FIRST TIME THROUGH THE UPWARDS CALCULATIONS
PROPCL	M	M = 1 FOR TOTALLY EXPENDABLE VEHICLE, M = 2 FOR EXPENDABLE UPPER STAGE ONLY, M = 3 FOR TOTALLY REUSABLE VEHICLE.
PROPCL	N	NUMBER OF CURRENT STAGE
PROPCL	NREQ	NUMBER OF STAGES REQUIRED. SHOULD EQUAL NTOT.

PROPCL	NTOT	NUMBER OF STAGES AVAILABLE TO VEHICLE
PROPCL	S	*SAFETY* FACTOR CONCERNING ENGINE THRUST. EQUALS 1 IF INPUT PERFORMANCE DATA IS PRACTICAL, > 1 IF INPUT DATA IS THEORETICAL. ALWAYS = 1 IN DORCA.
PROPCL	V	VELOCITY AVAILABLE FROM THRUST OF THIS STAGE
PROPCL	VIN	VELOCITY INCREMENT DELTA V REQUIRED FOR ORBIT CHANGE (LEG TRAVEL)
PROPCL	VRQ	VELOCITY REQUIRED OF STAGES NOT YET CALCULATED
PROPCL	WB	WB(I) = WEIGHT OF I-TH STAGE AT BURNOUT
PROPCL	WE	WEIGHT OF VEHICLE AT END OF BURNOUT OF STAGE N
PROPCL	WP	PROPELLANT REQUIRED FOR THIS STAGE TO ACHIEVE REQUIRED VELOCITY VRQ
PROPCL	WPLDN	WEIGHT OF PAYLOAD DOWN
PROPCL	WPLUP	PAYLOAD UP
PROPCL	WPMX	WPMX(I) = MAXIMUM USABLE PROPELLANT WEIGHT FOR STAGE I.
PROPCL	WPREQ	WEIGHT OF REQUIRED PROPELLANT
RDCONT	B	A CONSTANT BLANK (=1H) USED FOR CHECKING FOR SPACES
RDCONT	CAPAC	WEIGHT CAPACITY FOR CONTAINER EQUAL TO CARD(3)
RDCONT	CLASS	CARGO HANDLING CLASSIFICATION EQUAL TO CARD(7)
RDCONT	FOUR	CONSTANT OF VALUE 4.
RDCONT	I	SCRATCH VARIABLE
RDCONT	IERR	ERROR FLAG SET IN DECODING INPUT NUMERIC VALUE.
RDCONT	IV	SCRATCH VARIABLE EQUIVALENCE TO V
RDCONT	I2	SCRATCH VARIABLE
RDCONT	J	SCRATCH VARIABLE
RDCONT	NAME	UNIQUE CONTAINER NAME AS READ FROM CARD INPUT EQUIVALENT TO CARD(1)
RDCONT	NCRD	NO. OF CARDS READ EQUAL TO NCONT
RDCONT	ONE	CONSTANT OF VALUE 1.
RDCONT	PENAL	CONTAINER EMPTY WEIGHT.
RDCONT	TWO	CONSTANT OF VALUE 2.
RDCONT	V	THE VALUE OF VOLUME IN FLOATING POINT
RDCONT	VOLUME	INPUT VOLUME FACTOR IN CODED FORMAT.
RDCRG	B	A CONSTANT BLANK (=1H) USED FOR CHECKING FOR SPACES
RDCRG	CAT	A FLAG WHICH IS SET INDICATING THE CATEGORY OF THE SHIPMENT. (1-MATERIAL, 2-PERSONNEL, 3-FACILITY OR SATELLITE, 4-VEHICLE)
RDCRG	CATEG	CONTAIN CARD IMAGE OF WORD DENOTING ITEM CATEGORY EQUIVALENT TO CARD(8)
RDCRG	CLASS	CONTAINER CLASS (1-CREW CAPSULE, 2-BULK CONTAINER, 3-NONE (DISCRETE ITEM), 4-PROPELLANT TANK)
RDCRG	CONTN	CONTAINS THE NAME OF THE CONTAINER IN WHICH A CARGO WILL BE CARRIED IN. EQUIVALENT TO CARD(6)
RDCRG	DB	EQUIVALENCE TO DDB

RDCRG	DESCRP	DESCRIPTOR TO BE USED AS IDENTIFICATION IN THE PRINTOUTS. EQUIVALENT TO CARD(3)
RDCRG	DWNWT	DESCRIBES THE DOWN WEIGHT OF THE CARGO ELEMENT IF THE CARGO ELEMENT IS TO BE SHIPPED DOWN THE LEG. EQUIVALENT TO CARD(12)
RDCRG	EXPND	CONTAINS CARGO VOLUME IN (A6,A4) FORMAT EQUIVALENT TO CARD(14)
RDCRG	I	USED AS AN INDEX
RDCRG	IERR	ERROR FLAG
RDCRG	I2	USED AS AN INDEX
RDCRG	J	INDEX VARIABLE
RDCRG	LX	TEMPORARY VARIABLE
RDCRG	NAME	CONTAINS THE NAME OF THE CARGO ELEMENT TO BE USED LATER IN THE MISSION DATA. EQUIVALENT TO CARD(1)
RDCRG	NCRD	NO. LAST READ CARD OF CARGO ELEMENT TABLE EQUIVALENCED TO NCE
RDCRG	NXTWRD	POINTS TO THE NEXT WORD IN THE DYNAMIC DATA BLOCK WHERE NEXT ENTRY WILL BE STORED.
RDCRG	PKWRD	DUMMY VARIABLE USED FOR TEMPORARY STORAGE
RDCRG	PNTER	POINTER TO EITHER THE VEHICLE OR FACILITIES TABLE
RDCRG	UPWT	DESCRIBES THE UP WEIGHT OF THE CARGO ELEMENT IF THE CARGO ELEMENT IS TO BE SHIPPED UP THE LEG. EQUIVALENT TO CARD(10)
RDCRG	VOLUME	VOLUME FACTOR OF CURRENT CARGO ITEM
RDFAC	BLANK	EQUAL TO 1H USED TO CHECK FOR BLANKS
RDFAC	DEVEL	NON-RECURRING DEVELOPMENT COST
RDFAC	I	INDEX VARIABLE
RDFAC	IERR	ERROR FLAG FROM SUBROUTINE -VALUE- WHILE DECODING NUMERIC INPUT
RDFAC	J	INDEX VARIABLE
RDFAC	JL	SCRATCH VARIABLE USE TO CHECK FOR NEWLY OCCURRED ERROR
RDFAC	JSPD	POINTER TO SPREAD TABLE FOR DEVELOPMENT COST.
RDFAC	K	INDEX VARIABLE
RDFAC	KSPD	POINTER TO SPREAD TABLE FOR PRODUCTION COST.
RDFAC	MONE	EQUAL TO -1.
RDFAC	NCARD	NO. OF CARDS READ
RDFAC	NCRD	NUMBER OF FACILITY CARDS READ (EQUIVALENCED TO NCARD)
RDFAC	RPROD	RECURRING PRODUCTION COST
RDFAC	YRS	MAX YEARS FACILITY LIFE BEFORE REPLACEMENT
RDLEG	I	INDEX VARIABLE
RDLEG	I2	INDEX VARIABLE
RDLEG	J	INDEX VARIABLE
RDLEG	LCNGSH	YES OR NO INDICATING LONGSHORING CAPABILITY

		OR SINGLE, INDICATING ONLY ONE CARGO ITEM PER FLIGHT.
RDLEG	MILLE	VARIABLE WITH INTEGER NAME BUT CONTAINING THE VALUE 1000. IN FLOATING POINT
RDLEG	NAME	UNIQUE LEG NAME READ FROM CARD INPUT
		EQUIVALENT TO CARD(1)
RDLEG	NCRD	NO. OF CARDS READ
		EQUAL TO NLEG
RDLEG	ONE	CONSTANT SET TO 1.
RDLEG	SUCESS	NAME OF NEXT LOWER (SUCCESSOR) LEG.
RDLEG	VEHICL	NAME OF DEFUALT VEHICLE
		EQUAL TO CARD(7)
RDLEG	ZERO	CONSTANT SET TO 0.
RDMISS	A	SUM OF UP WEIGHTS OF COMPONENTS OF COUPLED ITEM
RDMISS	B	SUM OF DOWN WEIGHTS OF COUPLED COMPONENTS
RDMISS	C	SUM OF VOLUMES OF COUPLEC COMPONENTS
RDMISS	CE	ARRAY CONTAINING INDICES OF THE COMPONENTS OF EACH COUPLED ITEM
RDMISS	CNUMB	MULTIPLICITY OF CARGO ELEMENT (NUMBER OF UNITS SHIPPED)
RDMISS	DATE	DATE HOLDS THE MISSION DATA START ENTRY IN FLOATING POINT.
RDMISS	DATECI	DATECI HOLDS MISSION DATA UPDATE DATE IN FLOATING POINT.
RDMISS	DATEPI	DATEPI HOLDS MISSION DATA PREVIOUS IOC DATA IN FLOATING POINT.
RDMISS	DATESP	DATESP HOLDS THE MISSION DATA SUSTAINING STOP DATE IN FLOATING POINT.
RDMISS	DATEST	DATEST HOLDS THE MISSION DATA SUSTAINING START DATE IN FLOATING POINT.
RDMISS	I	INDEX VARIABLE
RDMISS	ICARGO	ARRAY OF 3 PACKED WORDS TO BE ENTERED INTO LEVEL I CARGO TABLE
RDMISS	ICE	NUMBER OF COMPONENTS IN CURRENT COUPLED CARGO ITEM
RDMISS	ICEN	POINTER TO LAST INDEX IN ARRAY CE TO DEFINE COMPONENTS OF COUPLED ITEM
RDMISS	ICE0	ICE0+1 POINTS TO FIRST INDEX IN ARRAY CE LISTING COUPLED COMPONENTS
RDMISS	ICF	ICF=1 INDICATES ROUTINE IS CURRENTLY PROCESSING A COUPLED ITEM. ICF=0 OTHERWISE.
RDMISS	IDATE	IDATE HOLDS THE MISSION DATA UPDATE DATE MINUS THE RELATIVE DATE
RDMISS	IDC	EQUATED TO DATECI
RDMISS	IDP	EQUATED TO DATEPI
RDMISS	IERR	ERROR FLAG FROM SUBROUTINE -VALUE-
RDMISS	IF	FIRST INPUT CARD WORD WHICH CONTAINS DESCRIPTOR
RDMISS	IL	LAST INPUT CARD WORD CONTAINING DESCRIPTOR
RDMISS	ILEG	ILEG HOLDS A POINTER TO CURRENT MISSION LEG IN LEG TABLE.
RDMISS	IPHASE	IPHASE HOLDS INTEGER CODE TO INDICATE PHASE OF MISSION DATA
RDMISS	IRTON	FLAG INDICATING MODE OF DELIVERY (0=DEPLOY, 1=RETRIEVE, 6=SERVICE,
		-1=DEFAULT)
RDMISS	ITEMP	ITEMP HOLDS AN INTEGER CODE TO INDICATE DISCRETE OR BULK CARGO.
		PROCESSING.
RDMISS	IVEH	IVEH = 0 - NO VEHICLE CARD
		1 - A VEHICLE IS SPECIFIED FOR EACH LEG
		2 - AT LEAST ONE LEG MUST BE CAPTURED
RDMISS	IWORD	PACKED WORD TO BE ADDED TO IFA OR IVA ARRAY

RDMISS	I1	I1 IS A CONSTANT VARIABLE EQUAL TO ONE
RDMISS	I2	LOOP LIMIT USED IN SEVERAL SENSES
RDMISS	J	INDEX VARIABLE
RDMISS	JCARGO	ARRAY OF 3 PACKED WORDS TO BE ENTERED INTO LEVEL I CARGO TABLE FOR COUPLED CARGO
RDMISS	JVEH	JVEH = 1 FOR REGULAR CARGO, JVEH = 2 FOR A DISCRETE CARGO TO BE CAPTURED WITH AN UP WEIGHT AND A DOWN WEIGHT.
RDMISS	JWORD	PACKED WORD USED TO FORM THIRD WORD OF LEVEL I DATA FOR COUPLED ITEM
RDMISS	K	INDEX VARIABLE
RDMISS	L	INDEX VARIABLE
RDMISS	LIOC	ARRAY OF IOC DATES FROM SCHEDULE CARD IN RDMISS
RDMISS	LNAME	CONTAINS CARGO NAME WHILE PROCESSING SCHEDULE AND SATELLITE CARDS.
RDMISS	LSCHD	ARRAY OF FLIGHTS/YEAR FOR SCHEDULING. TEMPORARY STORAGE WHEN SATELLITE CARD IS INPUT.
RDMISS	L3	SCRATCH VARIABLE
RDMISS	M	SCRATCH VARIABLE
RDMISS	MAX	MAX HOLDS A CONSTANT EQUAL TO THE NUMBER OF ENTRIES IN THE PROGRAM NAME TABLE.
RDMISS	MM	CARGO CATEGORY
RDMISS	MN	MISSION NUMBER FOR THIS CARGO
RDMISS	MUST	MUST HOLDS A FLAG TO DETERMINE IF A MISSION ENTRY FOLLOWS A PROGRAM ENTRY IN THE INPUT DATA.
RDMISS	M1	FIRST CELL IN CE CONTAINING INDEX FOR COMPONENTS OF THIS COUPLED ITEM
RDMISS	M2	LAST CELL IN CE CONTAINING INDEX FOR COMPONENTS OF THIS COUPLED ITEM
RDMISS	N	SCRATCH VARIABLE
RDMISS	NAME	NAME IS AN ARRAY WHICH HOLDS THE MISSION DATA CARD IMAGES. EQUIVALENT TO CARD(1)
RDMISS	NBASE	NBASE POINTS TO A PROGRAM NAME IN THE PROGRAM NAME TABLE.
RDMISS	NIOC	TEMPORARY VARIABLE FOR IOC DATE
RDMISS	NN	VEHICLE INDEX INSERTED INTO IVA TABLE ENTRY
RDMISS	NSCD	NUMBER OF SCHEDULE DATES STORED IN THE LIOC ARRAY
RDMISS	NSCHD	CONTAINS CODED WORD -SCHEDU- OR -SHIPMT- WHILE RDMISS IS PROCESSING SCHEDULES
RDMISS	NSHP	NUMBER OF VALUES OF SHIPMENT DATA STORED IN THE LSCHD ARRAY
RDMISS	NUMBER	MULTIPLICITY OF CARGO ELEMENT (NUMBER OF UNITS SHIPPED)
RDMISS	NWORD	NWORD HOLDS THE CODED LIST OF VEHICLES SHIPPED TO BE PUT INTO THE IVA ARRAY.
RDMISS	PHASE	PHASE HOLDS THE PHASE OF OPERATION ENTRY IN MISSION DATA.
RDMISS	PN	PROGRAM NUMBER FOR THIS CARGO (INTEGER)
RDMISS	PNMN	ARRAY WITH COMPOSITE PROGRAM/MISSION NUMBER (6 BITS EACH)
RDMISS	TEMP	TEMP IS A TEMPORARY BUFFER WHICH IMPLEMENTS THE FORMAT CONVERSION OF 18 CHARACTER MISSION AND PROGRAM NAMES.
RDMISS	WLF	ARRAY OF COUPLED COMPONENT WEIGHT LOAD FACTORS, IN INTEGER FORMAT. EQUIVANCED TO WTLF AND INDEXED THE SAME AS ARRAY CE. EACH FACTOR = 100000 * COMPONENT WEIGHT / TOTAL COUPLED ITEM WEIGHT.

RDMISS	WTDN	CARGO DOWN WEIGHT
RDMISS	WTLF	UP OR DOWN WEIGHT OF COUPLED COMPONENT TO BE USED IN COMPUTING WEIGHT LOAD FACTORS. FLOATING POINT FORMAT. WTLF IS AN ARRAY EQUIVALENCED TO WLF.
RDMISS	WTUP	CARGO UP WEIGHT
RDRPT	I	INDEX VARIABLE
RDSPD	BLANK	2-CELL ARRAY CONTAINING FIRST FIELD OF INPUT CARD. EQUIVALENT TO CARD(1-2).
RDSPD	FACT	ARRAY OF 1-5 COST SPREADING FACTORS ON INPUT CARD IN HOLLERITH FORMAT
RDSPD	FACTOR	ARRAY OF 1-7 COST SPREADING FACTORS ON INPUT CARD IN HOLLERITH FORMAT
RDSPD	I	INDEX
RDSPD	IERR	ERROR FLAG
RDSPD	IOC	SEQUENCE NUMBER OF THE SPREADING FACTOR THAT CORRESPONDS TO THE VEHICLE OR FACILITY IOC DATE. EQUIVALENT TO CARD(5)
RDSPD	IX	TEMPORARY VARIABLE
RDSPD	I2	INDEX
RDSPD	NAME	NAME OF SPECIFIC SPREADING FUNCTION EQUIVALENT TO CARD(1) AND BLANK
RDSPD	NCRD	NO. OF CARDS READ BY THIS SUBROUTINE
RDSPD	NXTWRD	NEXT WORD IN THE DYNAMIC DATA BLOCK IN WHICH THE NEXT ENTRY WILL BE STORED
RDSPD	NYR	NUMBER OF YEARS OVER WHICH SPREADING FACTORS APPLY. EQUIVALENT TO CARD(3) AND FACTOR
RDSPD	SPD	CONTAINS SUM OF SPREADING FACTORS.
RDSPD	XX	TEMPORARY VARIABLE
RDVEH	BLANC	EQUAL TO 1H FOR COMPARISON IN FINDING BLANKS
RDVEH	BLANK	EQUIVALENT TO FIRST 10 COLUMNS OF INPUT CARD TO TEST FOR BLANK FIELD.
RDVEH	DOWNWT	2-CELL ARRAY CONTAINING INPUT CODED REPRESENTATION OF DNMAX (MAXIMUM PAYLOAD VEHICLE CAN CARRY DOWNWARDS ON THIS LEG)
RDVEH	EXPWT	2-CELL ARRAY CONTAINING INPUT CODED REPRESENTATION OF EXPMAX (MAXIMUM PAYLOA VEHICLE CAN CARRY UPWARDS ON THIS LEG IF IT IS EXPENDED)
RDVEH	I	INDEX VARIABLE
RDVEH	ICPDDB	POINTER TO CURRENT WORD BEING FILLED IN DDB ARRAY
RDVEH	IDV	VELOCITY INCREMENT (DELTA V) FOR THIS LEG
RDVEH	IERR	ERROR FLAG FROM SUBROUTINE VALUE
RDVEH	IFOUR	CONTAINS FLOATING POINT CONSTANT 4.0 WITH INTEGER NAME
RDVEH	IPN	WHEN EXTRACTED FROM TBVEH TABLE, CONTAINS NUMBER OF CELLS IN DDB USED FOR DATA FOR THIS VEHICLE
RDVEH	IPO	LOCATION OF START OF DATA FOR THIS VEHICLE IN DDB.
RDVEH	ISP	CONTAINS CODED WORD ≠ISP≠
RDVEH	IVOL	VOLUME
RDVEH	I2	UPPER LIMIT OF DO LOOP EQUAL TO NVEH-1
RDVEH	J	INDEX VARIABLE
RDVEH	JX	FLAG SET TO 1 WHEN AT LEAST THE MINIMUM AMOUNT OF DATA REQUIRED HAS

		BEEN INPUT FOR THIS VEHICLE.
RDVEH	K	INDEX VARIABLE
RDVEH	L	INDEX VARIABLE
RDVEH	LEGNM	NAME OF LEG ON WHICH VEHICLE CAN BE USED.
RDVEH	LIFFLT	MAX NO. FLIGHTS BEFORE REPLACEMENT
RDVEH	LIFYRS	MAX YEARS OPERATION BEFORE REPLACEMENT
RDVEH	M	FLAG INDICATING EXPENDABLE/REUSABLE CHARACTERISTICS OF VEHICLE. SEE PERLINK. ALSO USED AS A LOOP VARIABLE
RDVEH	MAXPYR	MAX NO. FLIGHTS PER YEAR EQUAL TO CARD(5), SPREAD, UPWT
RDVEH	MINLD	MIN WEIGHT TO JUSTIFY FLIGHT EQUAL TO CARD(1), EXPWT, REFURB
RDVEH	NAME	VEHICLE NAME.
RDVEH	NCRD	NO. OF CARDS READ AS INPUT
RDVEH	NFLAG	NFLAG = 4 FOR DRY STAGES, = 5 FOR WET STAGES CARD.
RDVEH	NXTWRD	INCREMENTING INTEGER USED IN INDEX
RDVEH	PROPEL	PROPELLANT WEIGHT FOR COMPLETE FUELING OF VEHICLE EQUAL TO CARD(3),NREV,LEGNM
RDVEH	RFLT	EQUAL TO CARD(7), LIFFLT(1) AND RPROD(1) NOT IN USE
RDVEH	RPROD	RECURRING PRODUCTION COST.
RDVEH	SPPROD	NAME OF SPREAD FUNCTION TO BE USED FOR RECURRING PRODUCTION COST. NOT REFERENCED BY NAME BUT EQUIVALENCED TO CARD(9-10).
RDVEH	SPREAD	NAME OF SPREAD FUNCTION TO BE USED FOR NONRECURRING DEVELOPMENT COST. NOT REFERENCED DIRECTLY BUT EQUIVALENCED TO CARD(5-6).
RDVEH	TEMP	TEMPORARY VARIABLE USED FOR PACKING OF STAGE INDICES
RDVEH	UPWT	MAXIMUM UP WEIGHT FOR VEHICLE ON THIS LEG. NOT REFERENCED BY NAME BUT EQUIVALENCED TO CARD(5-6).
RDVEH	UPWT	2-CELL ARRAY CONTAINING INPUT CODED REPRESENTATION OF UPMAX (MAXIMUM PAYLOAD VEHICLE CAN CARRY UPWARDS ON THIS LEG IF NOT EXPENDED)
RDVEH	WPEXP	MAXIMUM PAYLOAD UP ON THIS LEG IF VEHICLE IS EXPENDED
RDVEH	WPLDN	MAXIMUM PAYLOAD DOWN ON THIS LEG
RDVEH	WPLUP	MAXIMUM PAYLOAD UP ON THIS LEG IF VEHICLE IS NOT EXPENDED
READER	BLANK	WORD CONTAINING CODED BLANK
READER	CHAR	ARRAY OF 80 CELLS, EACH CONTAINING ONE CHARACTER FROM CURRENT INPUT CARD.
READER	IP	INDEX IN TRVEH MATRIX OF VEHICLE JUST NAMED IN PREFERENCE LIST
READER	IPREF	IF IPREF=1, ROUTINE IS PROCESSING VEHICLE PREFERENCE LIST
READER	IRPT	NUMBER OF REPORT REQUESTS READ SO FAR.
READER	ITEST	WORD CONTAINING FIRST CHARACTER ONLY OF CURRENT CARD. IF BLANK AND IF LAST CARD WAS A COMMENT CARD, THEN THIS ONE IS, TOO.
READER	IY	FIRST YEAR (RELATIVE TO RLDATE) FOR PREFERENCE VEHICLE
READER	JY	LAST YEAR (RELATIVE TO RLDATE) FOR PREFERENCE VEHICLE
READER	NEXT	INDICATES WHERE TO START SHIFT OF CARD COLUMNS WHEN REFORMATTING INPUT CARDS WHICH CONTAIN = SIGNS

READER	NSW	FLAG SET TO ONE WHEN INPUT CARDS ARE TO BE PRINTED
READER	OFTN	LIST OF ACCEPTABLE OPTION NAMES
READER	OFTV	LIST OF ACCEPTABLE HOLLERITH VALUES FOR EACH OPTION
READER	YEAR	FIRST OR LAST YEAR (ABSOLUTE) FOR PREFERENCE VEHICLF
REPORT	JFLAG	FLAG INDICATING IF VEHICLE AND/OR COST REPORT IS TO BE PRINTED.
RO	WORD2 LE*	PACKED WORD 2 FOR SHIPPING LEVEL I CARGO ON LOWER LEGS
SORT	A	ANY MATRIX OF DATA TO BE SORTED. A(MM,NN)
SORT	I	INDEX VARIABLE
SORT	II	FIRST COLUMN TO BE SORTED (ALWAYS TAKEN AS 1)
SORT	IL	ARRAY OF LOWER BOUNDS OF SUBINTERVALS TO BE SAVED AND SORTED LATER.
SORT	IU	ARRAY OF UPPER ROUNDS OF SUBINTERVALS TO BE SAVED AND SORTED LATER.
SORT	J	INDEX VARIABLE
SORT	JJ	LAST COLUMN TO BE SORTED (ALWAYS TAKEN AS NN)
SORT	K	COLUMN POINTER.
SORT	L	COLUMN POINTER.
SORT	LL	ELEMENT NO. ON WHICH ROUTINE SORTS THE COLUMN
SORT	M	INDEX VARIABLE
SORT	MM	NO. OF ROWS IN MATRIX A
SORT	N	EQUAL TO LL
SORT	NN	NO. OF COLUMNS IN MATRIX A
SORT	NP	EQUAL TO MM
SORT	T	VARIABLE CONTAINING THE VALUE OF THE MEDIAN ELEMENT
SORT	TT	VARIABLE USED IN INTERCHANGING COLUMNS.
SORT	Z	INDEX VARIABLE
SPDAP	COUNT	NUMBER OF UNITS PURCHASED IN A GIVEN YEAR
SPDAP	I	INDEX VARIABLE
SPDAP	IYRIOC	YEAR OF SPREAD FUNCTION RANGE IN WHICH ITEM IS DELIVERED.
SPDAP	IYRSP	NUMBER OF YEARS SPANNED BY SPREAD FUNCTION
SPDAP	J	TEMPORARY VARIABLE
SPDAP	K	INDEX VARIABLE
SPDAP	L	TEMPORARY VARIABLE
SPDAP	M	TEMPORARY VARIABLE
SPDAP	Y	REAL VALUE OF L
SPRINT	ACP	SUBTOTAL OF FLIGHTS AND VOLUME/LOAD FACTORS FOR THIS VEHICLE/YEAR, ALL LEGS
SPRINT	ACT	GRAND TOTAL OF FLIGHTS AND VOLUME/LOAD FACTORS FOR THIS YEAR
SPRINT	BLANK	BLANK WORD
SPRINT	BLF	BULK LOAD FACTOR.
SPRINT	DNMAX	MAXIMUM VEHICLE CAPACITY DOWNWARDS
SPRINT	DOWN	DOWN WEIGHT OF CARGO ITEM.
SPRINT	EXP	CONTAINS WORD -EXP- IF VEHICLE IS EXPENDED, OTHERWISE BLANK
SPRINT	EXPMAX	MAXIMUM CAPACITY UPWARDS IF VEHICLE IS EXPENDED
SPRINT	EXPND	CONTAINS WORD -EXP-
SPRINT	I	INDEX VARIABLE

SPRINT	IA	COUNTER TO DETERMINE POSITION IN CARGO TABLE
SPRINT	IB	LOOP VARIABLE
~~SPRINT~~	~~IFLT~~	~~FIRST SET TO LEG/VEHICLE/YEAR/FLIGHT NUMBER OF CURRENT CARGO ITEM.~~
		LATER RESET TO PURE FLIGHT NUMBER.
SPRINT	II	NUMBER OF WORDS/LOGICAL RECORD ON TAPE CONTAINING CARGO TABLE
~~SPRINT~~	~~IJ~~	~~LOCATION OF DATA FOR THIS CARGO ITEM IN CARGO ELEMENT TABLE.~~
SPRINT	INDEX	COMPOSITE VEHICLE/LEG INDEX FOR XLF MATRIX. INDEX = 100*VEHICLE + LEG
SPRINT	IRTN	IRTN = 30 MEANS ENTIRE CARGO TABLE HAS BEEN PROCESSED
~~SPRINT~~	~~IV~~	~~LOOP VARIABLE~~
SPRINT	IYEAR	YEAR (4 DIGITS)
SPRINT	I1	PROGRAM NUMBER.
~~SPRINT~~	~~I2~~	~~MISSION NUMBER.~~
SPRINT	I3	LEG NUMBER
SPRINT	I4	VEHICLE NUMBER.
SPRINT	I5	RELATIVE DATE.
SPRINT	I6	FLIGHT NUMBER
SPRINT	I7	CARGO ELEMENT INDEX.
~~SPRINT~~	~~I8~~	~~DIRECTIONAL FLAG (0 = UP, 1 = DOWN)~~
SPRINT	J	INDEX VARIABLE
SPRINT	JFLT	LEG/VEHICLE/YEAR/FLIGHT NUMBER OF LAST CARGO ITEM.
~~SPRINT~~	~~JL~~	~~POINTS TO END OF DATA FOR THIS VEHICLE IN DDB~~
SPRINT	JX	NUMBER OF LINES ENTERED IN XLF ARRAY
SPRINT	JXMAX	MAXIMUM NUMBER OF VEHICLE/LEG COMBINATIONS WHICH MATRIX XLF CAN ACCOMODATE
~~SPRINT~~	~~J1~~	~~POINTS TO BEGINNING OF DATA FOR THIS VEHICLE IN DDB~~
SPRINT	K	LOOP VARIABLE
SPRINT	KK	NUMBER OF WORDS OF CARGO TABLE TO READ FROM TAPE
~~SPRINT~~	~~K1~~	~~STARTING POINT TO PLACE CARGO TABLE DATA IN DDB~~
SPRINT	LF	EQUIVALENCED TO XLF
SPRINT	LFA	LOAD FACTOR X 100000, IN INTEGER FORMAT.
~~SPRINT~~	~~LFB~~	~~BULK LOAD FACTOR X 100000, IN INTEGER FORMAT.~~
SPRINT	LNDEX	VALUE OF INDEX FOR LAST CARGO ITEM
SPRINT	LTH	CUTOFF VALUE OF YEARS TO PRINT IN HEADING
~~SPRINT~~	~~N~~	~~LENGTH OF DATA FOR THIS VEHICLE IN DDB~~
SPRINT	NC	LOOP VARIABLE FOR NUMBER OF PAGES (DEPENDS ON YEAR SPAN)
SPRINT	NYEAR	FIRST YEAR OF ENTIRE CASE.
~~SPRINT~~	~~TBL1~~	~~LEG NAME, FIRST 6 CHARACTERS~~
SPRINT	TBL2	LEG NAME, LAST 4 CHARACTERS
SPRINT	TLF	LOAD FACTOR OF CURRENT CARGO ITEM.
~~SPRINT~~	~~TLOAD~~	~~TOTAL OF LOAD FACTORS FOR ALL CARGO ITEMS ON THIS FLIGHT.~~
SPRINT	TOTAL	CONTAINS CODED WORD -TOTAL- FOR PRINTOUT
SPRINT	TRLF	LOAD FACTOR FOR THIS FLIGHT
~~SPRINT~~	~~TWTDW~~	~~TOTAL DOWN WEIGHT ON THIS FLIGHT.~~
SPRINT	TWTUP	TOTAL UP WEIGHT ON THIS FLIGHT.

SPRINT	UP	UP WEIGHT OF CARGO ITEM.
SPRINT	UPMAX	MAXIMUM VEHICLE WEIGHT CAPACITY UPWARDS
~~SPRINT~~	~~VOLF~~	~~VOLUME FACTOR FOR THIS FLIGHT~~
SPRINT	VOLMAX	VEHICLE VOLUME CAPACITY
SPRINT	VOLUME	TOTAL VOLUME ON FLIGHT
~~SPRINT~~	~~XLF~~	~~MATRIX TO ACCUMULATE FLIGHTS AND VOLUME/LOAD FACTORS FOR EACH VEHICLE/LEG~~
TABLES	IBVEH	POINTER TO BEGINNING OF DATA FOR THIS VEHICLE IN DDB
TABLES	ILVEH	POINTER TO END OF DATA FOR THIS VEHICLE IN DDB
~~TABLES~~	~~IWVEH~~	~~NUMBER OF WORDS OF DATA FOR THIS VEHICLE IN DDB~~
TABLES	LEN	NUMBER OF WORDS OF DATA FOR THIS SPREAD FUNCTION IN DDB
TABLES	LIM	NUMBER OF ENTRIES IN VEHICLE (R FACILITY ACQUISITION TABLE, WHICHEVER IS GREATER
~~TABLES~~	~~NB~~	~~BEGINNING OF DATA FOR THIS SPREAD FUNCTION IN DDB~~
TABLES	NL	END OF DATA FOR THIS SPREAD FUNCTION IN DDB
TRAFIC	DVTT	DETAILED VEHICLE TRAFFIC TABLE. CONTAINS FLIGHTS IN EACH YEAR FOR EACH PHYSICAL VEHICLE PLUS A STATUS FLAG IN COLUMN 32 (0-NOT YET USED, 1-ACTIVE, 2-RETIRED, 3-TO BE EXPENDED)
TRAFIC	FLTAC	MATRIX OF VEHICLES ACQUIRED IN EACH YEAR
~~TRAFIC~~	~~FLTAC~~	~~A MATRIX OF VEHICLE ACQUISIONS BY VEHICLE TYPE AND BY YEAR....~~
TRAFIC	IDATE	REFERENCE DATE (1970). SAME AS RLDATE
TRAFIC	IE1	COUNTER ON EXPENDED FLIGHTS ACCOUNTED FOR SO FAR
~~TRAFIC~~	~~II~~	~~LOCATION OF SECOND PART OF VEHICLE NAME, FOR PRINTOUT. ACCOMPANIES LVT.~~
TRAFIC	IPO	LOCATION OF START OF INPUT DATA FOR THIS VEHICLE IN DDB
TRAFIC	IRTN	INDICATES WHETHER NEWLY ACTIVATED VEHICLE WILL BE EXPENDED THIS YEAR ~~(IRTN=2) OR NOT (IRTN=1)~~
TRAFIC	ITEMP	TEMPORARY VARIABLE USED IN EXTRACTING DATA FROM CARGO ELEMENT TABLE
TRAFIC	IV	INDEX OF VEHICLE TYPE CURRENTLY BEING SCHEDULED
~~TRAFIC~~	IVAQ	~~NUMBER OF VEHICLES OF THIS TYPE ACQUIRED THIS YEAR REMAINING IN~~ UNDETERMINED STATUS
TRAFIC	IVAT	TOTAL COUNT OF VEHICLES OF THIS TYPE ACQUIRED IN ALL YEARS
~~TRAFIC~~	~~IVEH~~	~~VEHICLE INDEX FOR LOWER LEGS~~
TRAFIC	IVTL	NUMBER OF VEHICLE CURRENTLY BEING ASSIGNED FLIGHTS. POINTS TO DVTT MATRIX AND NFR ARRAYS
~~TRAFIC~~	~~IX~~	~~CONTAINS NUMBER OF EXPENDED VEHICLES IN LEFT HALF OF WORD (BITS 0-17)~~
TRAFIC	IYR	YEAR IN WHICH FLIGHTS ARE ASSIGNED, RELATIVE TO IDATE = RLDATE
TRAFIC	I1	INDEX FOR UNPACKING STAGES WHEN MORE VEHICLES ARE ACQUIRED AND SHIPPED IN ~~CARGO TABLE~~
TRAFIC	I2	BIT POSITION FOR UNPACKING STAGES
TRAFIC	JJ	YEAR IN WHICH EXTRA VEHICLE IS NEEDED (4 DIGITS FOR PRINTOUT)
~~TRAFIC~~	~~JYR~~	~~YEAR IN WHICH ACQUIRED VEHICLE IS ADDED TO PHASE I CARGO TABLE~~
TRAFIC	KOUNT	COUNT ON NUMBER OF PHYSICAL VEHICLES PRINTED SO FAR. DIFFERS FROM IVTL, WHICH IS REDUCED WHEN VEHICLES ARE EXPENDED OR RETIRED
~~TRAFIC~~	~~L~~	~~INDICATES PROPER SLOT FOR YEAR NY1 IN MTT/FLTAC ARRAYS~~
TRAFIC	LINE	ARRAY CONTAINING ANNUAL COUNTS OF VEHICLES ACQUIRED TO DATE

TRAFIC	LVT	LOCATION OF START OF INPUT DATA FOR THIS VEHICLE IN DDB
TRAFIC	MAXPY	MAXIMUM FLIGHTS/YEAR FOR THIS TYPE VEHICLE
TRAFIC	MAXVEH	MAXIMUM NUMBER OF PHYSICAL VEHICLES OF ANY TYPE WHICH PROGRAM CAN ACCOMODATE (100)
TRAFIC	MNYRS	LIFETIME (YEARS) OF THIS TYPE VEHICLE
TRAFIC	MTF	MAXIMUM TOTAL FLIGHTS IN LIFETIME OF THIS TYPE VEHICLE
TRAFIC	MTT	MASTER TRAFFIC TABLE. INITIALLY CONTAINS NUMBER OF FLIGHTS AND NUMBER OF EXPENDED FLIGHTS FOR EACH TYPE VEHICLE IN EACH YEAR. LATER, THE EXPENDED COUNTS ARE REMOVED.
TRAFIC	MXVH1	UPPER INDEX FOR DO-LOOP PURGING A RETIRED OR EXPENDED VEHICLE
TRAFIC	N	CARGO ELEMENT TABLE INDEX CORRESPONDING TO VEHICLE INDEX IVEH
TRAFIC	NBASE	LOCATION OF DATA IN CARGO ELEMENT TABLE
TRAFIC	NEWLEG	LEG OF CURRENT INTEREST, IF TRAFIC WAS CALLED FROM LEGPRO. NOT USED WHEN TRAFIC IS CALLED FROM REPORT.
TRAFIC	NEXTRA	NUMBER OF EXTRA FLIGHTS IN LIFETIME OF THIS VEHICLE WHICH WILL REMAIN UNUSED IF VEHICLE IS USED AT ITS MAXIMUM RATE IN ITS REMAINING YEARS AFTER CURRENT YEAR.
TRAFIC	NE1	NUMBER OF EXPENDED FLIGHTS
TRAFIC	NFEXP	NUMBER OF FLIGHTS REMAINING FOR A VEHICLE TO BE EXPENDED
TRAFIC	NFLTS	NUMBER OF FLIGHTS STILL UNASSIGNED IN THIS YEAR FOR THIS TYPE VEHICLE
TRAFIC	NFR	ARRAY CONTAINING NUMBER OF FLIGHTS REMAINING IN LIFETIME OF EACH VEHICLE ENTERED IN DVTT.
TRAFIC	NFTBA	NUMBER OF FLIGHTS TO BE ASSIGNED TO CURRENT VEHICLE
TRAFIC	NF1	NUMBER OF FLIGHTS EXTRACTED FROM NFTBL ARRAY. ALSO, NUMBER OF ACQUIRED VEHICLES LIST IN VEHICLE ACQUISITION TABLE
TRAFIC	NL1	LEG INDEX OF ACQUIRED VEHICLE(S)
TRAFIC	NRV	NUMBER OF REQUIRED/REMAINING VEHICLES FOR THIS YEAR
TRAFIC	NVIF	NUMBER OF VEHICLES IN FLEET IN CURRENT YEAR
TRAFIC	NVMAX	POINTER TO LAST ENTRY IN MTT AND FLTAC MATRICES
TRAFIC	NV1	VEHICLE INDEX OF ENTRY IN IVA/NFTBL ARRAYS
TRAFIC	NYA	YEAR IN WHICH THIS VEHICLE WAS ACQUIRED
TRAFIC	NYACT	NUMBER OF YEARS THIS VEHICLE HAS BEEN ACTIVE, INCLUDING CURRENT YEAR
TRAFIC	NYLEFT	NUMBER OF YEARS REMAINING IN LIFETIME OF THIS VEHICLE AFTER THIS YEAR.
TRAFIC	NY1	DATE OF ENTRY IN NFTBL/IVA ARRAYS
TRAFIC	TOTFLT	TOTAL NUMBER OF VEHICLES AVAILABLE IN FLEET IN EACH YEAR
TRAFIC	VEH1	FIRST OF TWO PACKED WORDS USED TO ENTER NEWLY ACQUIRED VEHICLES INTO PHASE I CARGO TABLE.
TRAFIC	VEH2	SECOND OF TWO PACKED WORDS USED TO ENTER NEWLY ACQUIRED VEHICLES INTO PHASE I CARGO TABLE.
VALUE	A	2 CELL ARRAY CONTAINING CODED VALUE IN (A6,A4) FORMAT
VALUE	C	A 19 CELLED STOAGE OF EACH CHARACTER IN THE CODED VALUE A
VALUE	CL	10 CELLED STORAGE OF VALES 0-9 FOR COMPARISON
VALUE	I	INDEX VARIABLE

```
VALUE       IERR       ERROR FLAG
                       IF IERR =  0            NO ERROR
                                 -1            COMPLETELY BLANK FIELD
                          1,2,...,10 POSITION OF ILLEGAL CHARACTER,
                                                MULTIPLE DECIMAL POINTS, OR
                                                EMBEDDED BLANKS.
VALUE       J          INDEX VARIABLE
VALUE       NDP        NO. OF DECIMAL POINTS
VALUE       NF         POSITION OF FIRST NON-BLANK CHARACTER
VALUE       NL         POSITION OF LAST  NON-BLANK CHARACTER
VALUE       N1         N1 = NL - 9          INDEX VARIABLE
VALUE       P          VALUE IN CODED FORMAT BUT RIGHT-ADJUSTED IN FIELD OF 10 CHARACTERS.
VALUE       V          CONVERTED VALUE IN FLOATING POINT
VEHLDF      FACT       NAME OF LOAD FACTOR
VEHLDF      I          INDEX
VEHLDF      IFLG       FLAG INDEX
VEHLDF      IM         MISSION NAME INDEX
VEHLDF      IP         PROGRAM NAME INDEX
VEHLDF      IV         VEHICLE NAME INDEX
VEHLDF      IYR        YEAR INDEX
VEHLDF      L          INDEX
VEHLDF      LDF        NAME OF LOAD FACTOR
VEHRPT      I          INDEX PARAMETER
VEHRPT      IFAC       DVTT(199,1) EQUIVALENCE TO IFAC(1)
VEHRPT      IFLG       FLAG INDEX
VEHRPT      IMIS       INDEX TO THE MISSION NAME IN THE MISSION TABLE.
VEHRPT      IPRO       INDEX TO THE PROGRAM NAME IN THE PROGRAM TABLE.
VEHRPT      IVEH       INDEX TO THE VEHICLE NAME IN THE VEHICLE TABLE.
VEHRPT      IVTOT      = VOTOT(1,1)
VEHRPT      J          INDEX VARIABLE
VEHRPT      K          INDEX VARIABLE
VEHRPT      NB         POINTS TO THE BEGINNING LOCATION OF A SET OF DATA FOR A VEHICLE IN THE
                       CARGO TABLE.
VEHRPT      NFR        EQUIVALENCE TO NFTBL
VEHRPT      NL         POINTS TO THE LAST LOCATION OF THE SET OF DATA FOR A VEHICLE IN THE
                       DYNAMIC DATA BLOCK.
VEHRPT      NV         NUMBER OF VEHICLES IN CARGO TABLE
VEHRPT      VDATA      CONTAINS DATA EXTRACTED FROM CARGO TABLE PERTAINING TO EACH VEHICLE.
VEHRPT      VTOT       SUM OF LOAD FACTORS FOR ALL FLIGHTS, GIVEN BY VEHICLE AND YEAR
/ /         DDB        DYNAMIC DATA BLOCK, CONTAINING CARGO TABLES AND MOST INPUT DATA.
                       HAS DUMMY DIMENSION OF 1 CELL IN ALL SUBROUTINES, BUT THE DIMENSION
                       OF ARRAY CE IN BLANK COMMON DEFINED IN MAIN ROUTINE DORCA ADDS 35000
                       MORE CELLS. THIS FACILITATES CHANGING THE EFFECTIVE SIZE OF DDB BY
```

		SIMPLY CHANGING THE VALUE OF VARIABLE LOWCOR AND THE DIMENSION OF CE IN ROUTINE DORCA ONLY.
/ASDAT/	CCAP	UNUSED CONTAINER CAPACITY (DOES NOT DEPEND ON DIRECTION)
/ASDAT/	CD	CD(I,J) TOTAL AMOUNT OF BULK CARGO STILL UNASSIGNED IN DIRECTION I (I = 1,2) FOR CONTAINER TYPE J(J = 1,....,NCONT).
/ASDAT/	CD2	CD2(I,J) = TOTAL WEIGHT OF ALL BULK CARGO FROM CAPTURE BIN OF CONTAINER TYPE J STILL UNASSIGNED IN DIRECTION I.
/ASDAT/	CR	TABLE OF CONTAINERS REQUISITIONED
/ASDAT/	DIRECT	DIRECTION OF TRAVEL INDICATOR 1, UP: AWAY FROM EARTH 2, DOWN: TOWARDS EARTH
/ASDAT/	DVTT	FOR A GIVEN TYPE OF VEHICLE, A MATRIX OF FLIGHT ASSIGNMENTS BY VEHICLE NUMBER AND BY YEAR.
/ASDAT/	EXP	CONTAINS CODED YES OR NO WORD IDICATING WHETHER CURRENT FLIGHT IS SCHEDULED TO BE EXPENDABLE
/ASDAT/	FACTOR	FACTOR(I) CONTAINS EQUIVALENT UP-WEIGHT FACTOR FOR A VEHICLE UP FACTOR(1) = 1.0, DOWN FACTOR(2) = UPMAX/DNMAX
/ASDAT/	FLTA	TABLE OF CARGO FLIGHT ASSIGNMENTS SCHEDULED FOR THIS LEG/VEHICLE/YEAR
/ASDAT/	ICT	INDICATES KIND OF CONTAINER REQUIRED BY CURRENT BULK CARGO ITEM
/ASDAT/	INDEX	IF NONZERO UPON RETURN FROM SUBROUTINE -FIND-, A CARGO ITEM SATISFYING REQUIRED MODE, TYPE, DIRECTION, CONTAINER TYPE, VOL. AND WT. LIMIT WAS FOUND AND ASSIGNED TO CURRENT FLIGHT. IF 0, NO SUCH ITEM WAS FOUND.
/ASDAT/	ITEMS	LOWER INDEX LIMIT ON UNASSIGNED ITEMS REMAINING IN WS/VOL MATRICES. ACTUAL NO. OF ITEMS REMAINING UNASSIGNED IS MAXI+1-ITEMS
/ASDAT/	JDONE	JDONE IS SET TO 1 WHEN ALL CARGO (UNCLUDING CAPTURE BIN) HAS BEEN ASSIGNED
/ASDAT/	LIMOCC	LIMOCC(I) IS THE MAXIMUM NUMBER OF CARGO ITEMS WHICH MAY BE ASSIGNED TO ANY FLIGHT UPWARDS (I=1) OR DOWNWARDS (I=2)
/ASDAT/	MANNED	INDICATES MANNED CAPSULE ASSIGNED THIS FLIGHT - 1, UP;2, DOWN 0, IF NO MANNED CAP. ASSIGNED
/ASDAT/	MCBULK	IF NONZERO, INDICATES THAT A CREW CAPSULE WAS ASSIGNED TO THIS FLIGHT AND THAT BULK CARGO HAS NOT YET BEEN LOADED INTO THE CAPSULE FREE SPACE.
/ASDAT/	MCHG	IF NONZERO, INDICATES THAT THE REMAINDER OF A BULK CARGO ITEM (OF WHICH A PORTION WAS JUST ASSIGNED) WAS REMOVED FROM THE CAPTURE BIN AND REDESIGNATED AS REGULAR CARGO.
/ASDAT/	MCT	INDEX IN CD AND TRCONT TABLE INDICATING WHAT KIND OF BULK CARGO IS TO BE LOADED, IF MORE THAN ONE KIND EXIST
/ASDAT/	MODE	CLASSIFICATION OF CARGO ITEMS ACCORDING TO TREATMENT MODE = 1 - CARGO REQUIRES EXPENDED VEHICLE. 2 - CARGO IS FROM CAPTURE BIN. 3 - CARGO IS REGULAR ITEM.
/ASDAT/	MTT	NAME OF THE MASTER TRAFFIC TABLE
/ASDAT/	NASS	NO. OF ASSIGNMENTS STORED IN FLTA FOR THIS VEHICLE/LEG/YEAR
/ASDAT/	NCALL	NUMBER OF CONSECUTIVE UNSUCCESSFUL CALLS TO SUBROUTINE -FIND-
/ASDAT/	NCR	NUMBER OF ENTRIES IN CONTAINER REQUISITION LIST CR.

```
/ASDAT/    NEXP      NO. OF UNASSIGNED CARGO ITEMS WHICH REQUIRE EXPENDABLE VEHICLE
/ASDAT/    NFLT      NO. FLIGHTS PER VEHICLE PER LEG PER YEAR
/ASDAT/    NFR       NUMBER OF FLIGHTS REMAINING FOR A VEHICLE IN A FLEET OF A GIVEN TYPE
                     OF VEHICLES.
/ASDAT/    NMODE     NMODE(I) = NUMBER OF CARGO ITEMS OF MODE I STILL UNASSIGNED.
                               I=1 FOR CARGO REQUIRING AN EXPENDED VEHICLE
                               I=2 FOR ITEMS IN THE CAPTURE BIN
                               I=3 FOR REGULAR CARGO ITEMS
/ASDAT/    NOCC      NOCC(I) IS THE NUMBER OF ITEMS ALREADY ASSIGNED TO CURRENT FLIGHT
                     UPWARDS (I=1) AND DOWNWARDS (I=2). A CONTAINER PLUS ALL ITS CONTENTS COUNTS
                     AS A SINGLE ITEM FOR THIS PURPOSE.
/ASDAT/    NPRI      IF NPRI=1, DEBUGGING INFORMATION OF THE ASSIGNMENT PROCEDURE IS PRINTED IN
                     SUBROUTINES ASINER AND FIND.
/ASDAT/    RVOL      REMAINING VEHICLE VOLUME CAPACITY ON CURRENT FLIGHT IN UP DIRECTION
                     (I=1) AND DOWN DIRECTION (I=2).
/ASDAT/    TOTFLT    FOR A GIVEN TYPE OF VEHICLE, A RUNNING TOTAL OF FLTAC BY YEAR
/ASDAT/    TRIFLE    CUTOFF VALUE TO DETERMINE WHEN CERTAIN QUANTITIES HAVE BEEN EXHAUSTED.
                     EQUAL TO 0.999 LB.
/ASDAT/    TW        TW(I,J) GIVES THE TOTAL WEIGHT TO BE CARRIED BY TH VEHICLE ON FLIGHT
                     NO. J IN DIRECTION I (1-UP, 2-DOWN)
/ASDAT/    TYPE      CLASSIFICATION OF CARGO ITEMS BY STORAGE REQUIREMETNTS
                     1- CREW,  2-BULK CARGO, 3-DISCRETE (SELF-CONTAINED) ITEM.
/ASDAT/    VCAP      AMOUNT OF VEHICLE CARRYING CAPACITY UNUSED,
                     EXPRESSED AS EQUIVALENT UP-WEIGHT
/ASDAT/    VDONE     IF VDONE=1, CURRENT VEHICLE FLIGHT IS FILLED UP.
/ASDAT/    VOL       VOLUME OF UNASSIGNED CARGO ITEM I
                     THIS ARRAY ACCOMPANIES MATRIX WS AND IS INDEXED THE SAME.
/ASDAT/    VOLMAX    MAX. VOLUME CAPACITY OF CURRENT CARGO
/ASDAT/    VTOT      CONTAINS SUMMARY DATA EXTRACTED FROM CARGO TABLE PERTAING TO EACH
                     VEHICLE
/ASDAT/    V1        FIRST PART OF CURRENT VEHICLE NAME
/ASDAT/    V2        SECOND PART OF CURRENT VEHICLE NAME
/ASDAT/    WLEFT     WLEFT(I,J,K) = TOTAL WEIGHT OF STILL-UNASSIGNED CARGO OF
                     MODE I, TYPE J, DIRECTION K.
/ASDAT/    XL1       FIRST PART OF CURREN LEG NAME
/ASDAT/    XL2       SECOND PART OF CURRENT LEG NAME
/CONT/     NCONT     NUMBER OF CONTAINERS INPUT
/CONT/     NCTMAX    MAXIMUM NUMBER OF CONTAINERS WHICH CAN BE INPUT
/CONT/     TBCONT    MATRIX CONTAINING 8 WORDS OF DATA FOR EACH CONTAINER
                          WORDS 1-2  CONTAINER NAME
                                3    CAPACITY (LBS)
                                4    CONTAINER WEIGHT (LBS)
                                5    CLASSIFICATION (1-CREW, 2-BULK, 4-PROPELLANT)
```

	6	VOLUME
	7	USED AS A FLAG IN SUBROUTINE CNTRPT
	8	RETURN/EXPEND OPTION
/CRGS/	NBCE	POINTS TO THE BEGINNING OF THE CARGO ELEMENT TABLE IN THE DYNAMIC DATA BLOCK.
/CRGS/	NCE	NUMBER OF CARGO ELEMENTS IN THE CARGO ELEMENT TABLE.
/CRGS/	NLCE	POINTS TO END OF CARGO ELEMENT TABLE IN DDB.
/CRGS/	NWCE	NWCE HOLDS THE NUMBER OF WORDS PER CARGO ELEMENT TABLE ENTRY.
/DDBS/	NCDB	NUMBER OF LEVEL 2 CARGO ITEMS REMAINING ON TAPE
/DDBS/	NBDDB	POINTS TO THE BEGINNING OF THE CARGO TABLE IN THE DYNAMIC DATA BLOCK.
/DDBS/	NBLK	NUMBER OF 510-WORD BLOCKS OF CELLS AVAILABLE IN DDB TO READ IN LEVEL 2 DATA
/DDBS/	NDDB	NUMBER OF CARGO SHIPMENTS
/DDBS/	NLDDB	LOCATION OF END OF PHASE II CARGO TABLE IN DDB.
/DDBS/	NWDDB	NUMBER OF WORDS PER ITEM IN PHASE II CARGO TABLE (EQUALS 3).
/FACS/	NBFAC	LOCATION, IN THE DYNAMIC DATA BLOCK, WHERE THE FACILITY TABLE BEGINS.
/FACS/	NFAC	NUMBER OF FACILITY ENTRIES IN THE FACILITY TABLE.
/FACS/	NLFAC	LOCATION OF END OF FACILITY DATA IN DDB ARRAY.
/FACS/	NWFAC	NUMBER OF WORDS FOR EACH FACILITY ENTRY.
/FTBL/	MTBL	MAXIMUM NUMBER OF ENTRIES IN NFTBL ARRAY
/FTBL/	NFTBL	FLIGHT TABLE. EACH ENTRY IS A PACKED WORD CONTAINING VEHICLE NUMBER, YEAR, NUMBER OF FLIGHTS, AND NUMBER OF EXPENDED VEHICLES.
/FTBL/	NTBL	NO. OF ITEMS IN NFTBL TABLE
/FTBL/	NTBL1	LOWER INDEX OF SUBSET OF NFTBL TO BE PROCESSED BY SUBROUTINE -TRAFIC-
/FTBL/	NTBL2	UPPER INDEX OF SUBSET OF NFTBL TO BE PROCESSED BY SUBROUTINE -TRAFIC-
/IFA/	IFA	FACILITY ACQUISITION LIST
/IFA/	NIFA	NUMBER OF ENTRIES IN FACILITY ACQUISITION LIST
/IVA/	IVA	VEHICLE ACQUISITION TABLE
/IVA/	MIFA	MAXIMUM LENGTH OF IFA ARRAY
/IVA/	MIVA	MAXIMUM LENGTH OF IVA ARRAY
/IVA/	NIVA	NUMBER OF ENTRIES IN VEHICLE ACQUISITION LIST
/KEL/	KEL	NUMBER OF THE MATRIX ELEMENT UPON WHICH THE SORT/MERGE IS PERFORMED
/LEGS/	LASTLG	POINTER TO LAST (SPECIAL) SLOT IN LEG TABLE
/LEGS/	NLEG	NO. OF ENTRIES IN LEG TABLE
/LEGS/	NLGMAX	MAX. DIMENSION OF THE LEG TABLE SET TO 31 IN DORCA
/LEGS/	TBLEG	TABLE CONTAINING ALL LEG DATA
/MAXS/	MAXA	MAXIMUM NUMBER OF ENTRIES IN FLTA MATRIX (ASSIGNED ITEMS)
/MAXS/	MAXC	MAXIMUM NUMBER OF ENTRIES IN CR LIST (REQUISITION CONTAINERS)
/MAXS/	MAXF	MAXIMUM NUMBER OF FLIGHTS FOR THIS VEHICLE, LEG AND YEAR
/MAXS/	MAXI	MAXIMUM NUMBER OF ENTRIES IN WS MATRIX (UNASSIGNED ITEMS)
/MISC/	CARD	16-CELL ARRAY CONTAINING IMAGE OF LAST CARD READ.
/MISC/	ICNT	NUMBER OF LEVEL II CARGO TABLE ITEMS IN CORE (NOT YET ON TAPE) AT THE MOMENT
/MISC/	IFLAG	A FLAG USE IN SPECIFYING REPORTS TO BE PRINTED

```
                     IFLAG(1)  = 1  SPRINT          IFLAG(6) = 1   COST
                     IFLAG(2)  = 1  CONTAINER       IFLAG(7) = 1   TABLES
                     IFLAG(3)  = 1  FACILITY        IFLAG(8) = 1   DEBUG
                     IFLAG(4)  = 1  TRAFFIC         IFLAG(9) = 1   CALVEH
                     IFLAG(5)  = 1  VEHICLE         IFLAG(10) = 1  PRTCAL
                     IFLAG(5) = 2 OR IFLAG(6) = 2 FOR SHORT REPORTS.
/MISC/    IPLTP      LOGICAL NUMBER OF PLOT TAPE
/MISC/    IPROP      THE NO. OF THE CARGO ELEMENT THAT CARRIES ALL THE PROPELLANT
                     IPROP = JPROP  +  ITBCNT
/MISC/    ITBCNT     THE NO. OF THE LAST CARGO ELEMENT IN CARGO ELEMENT TABLE
                     BEFORE THE CONTAINERS ARE ADDED
/MISC/    I77        OCTAL CONSTANT 777777777777 USED FOR PACKING. SET IN SUBR. RDVEH
/MISC/    JERR       VARIABLE USED TO COUNT THE NUMBER OF ERRORS THAT OCCUR.
/MISC/    JFLAG      JFLAG=1 ON FIRST PASS THRU COST REPORT (NO PRINTING)
                          =0 ON SECOND PASS (PRINTOUT IS GENERATED)
/MISC/    JPROP      NO. OF THE LAST CONTAINER THAT CAN CARRY A PROPELLANT TANK
/MISC/    J77        OCTAL CONSTANT 000000000077 USED FOR PACKING. SET IN SUBR. LEGPRO
/MISC/    KOPT       ARRAY CONTAINING CURRENT STATUS OF INPUT OPTIONS
                          WORD 1 - PROPELLANT OFF-LOADING/FULL TANK
                               2 - SPACE/GROUND BASED
                               3 - SINGLE/MULTIPLE CARGO DEPLOYMENT
                               4 - MANUAL/AUTOMATIC VEHICLE DELIVERY
                               5 - CONTAINER RETURN/EXPEND
                               6 - CARGO FOR CAPTURE BIN/ASSIGNED VEHICLE
/MISC/    LINE       A LINE OF DATA WHICH SERVES AS A TEMPORARY STORAGE FOR A PRINTED LINE
                     OF OUTPUT
/MISC/    LOWCOR     NUMBER OF CELLS IN DDB ARRAY (SET IN DORSA)
/MISC/    L2WOT      LOGICAL TAPE NUMBER FOR WRITE-OUT OF LEVEL II CARGO TABLE
/MISC/    MAXCGN     TOTAL NUMBER OF COMPOSITE CARGO GROUPS
/MISC/    MWPL       NUMBER OF WORDS/LINE ON LAST PAGE OF COST REPORT (PAGE NR)
/MISC/    NAFAC      INDICATES TO SUBROUTINES PDCRG AND RDFAC HOW TO PROCESS INPUT CARDS.
                     NAFAC=0 - PROCESS ALL CARDS UNTIL A *TABLE* CARD IS ENCOUNTERED.
                     NAFAC≠0 - PROCESS ONLY ONE CARD, THEN RETURN TO CALLING ROUTINE.
                               (1-FACILITY CARD, 2-CARGO ELEMENT, 3-SCHEDULE, 4-SATELLITE)
/MISC/    NFW        FIRST ELEMENT OF INTEREST IN COST1, ETC., ARRAYS
/MISC/    NLW        LAST  ELEMENT OF INTEREST IN COST1, ETC., ARRAYS
/MISC/    NOSC       NUMBER OF CELLS NEEDED IN DDB TO STORE 10 MORE CARGO ELEMENTS.
/MISC/    NOSF       NUMBER OF CELLS NEEDED IN DDB TO STORE DATA FOR 10 MORE FACILITIES.
/MISC/    NR         NUMBER OF PAGES REQUIRED FOR EACH COST REPORT (ALSO, NUMBER OF SCRATCH
                     TAPES)
/MISC/    NVMAX      INDEX OF LAST VEHICLE ACTUALLY USED. NVMAX ≤ NVEH.
/MISC/    NWPL       NUMBER OF WORDS/LINE ON COST REPORT, PAGE 1,2,...,NR-1.
/MISC/    PROPUP     CAPACITY OF PROPELLANT TANK
```

/MISS/	LNGTH	NUMBER OF CARGO ITEMS IN PHASE I CARGO TABLE
/MISS/	NBMISS	LOCATION OF BEGINNING OF PHASE I CARGO TABLE IN DDB.
/MISS/	NLMISS	LOCATION OF END OF PHASE I CARGO TABLE IN DDB
/MISS/	NWMISS	NWMISS HOLDS THE NUMBER OF WORDS PER MISSION DATA ENTRY.
/PROG/	MNAME	MISSION NAME TABLE
/PROG/	NMISS	NMISS HOLDS THE NUMBER OF ENTRIES IN THE MISSION NAME TABLE.
/PROG/	NPROG	NPROG HOLDS THE NUMBER OF ENTRIES IN THE PROGRAM NAME TABLE.
/PROG/	PNAME	PNAME ARRAY HOLDS MISSION NAMES PROVIDED BY MISSION DATA INPUT.
/SEG32/	NFILE	NUMBER OF 510-WORD RECORDS WHICH CAN BE ACCOMODATED IN AVAILABLE SPACE
/SPDS/	NPSPD	POINT TO THE BEGINNING LOCATION OF THE SPREAD TABLE IN THE DYNAMIC DATA BLOCK.
/SPDS/	NLSPD	POINTS TO THE LAST ENTRY OF THE SPREAD TABLE IN THE DYNAMIC DATA BLOCK.
/SPDS/	NSPD	NUMBER OF SPREAD FUNCTIONS
/SPDS/	TBSPD	SPREAD TABLE NAME (LOCATION)
/SRTMOD/	MODE	DETERMINES WHETHER SORT IS ALPHANUMERIC (MODE=0) OR ALGEBRAIC (MODE≠0)
/VEHS/	NBVEH	LOCATION OF FIRST ENTRY OF VEHICLE TABLE IN DDB
/VEHS/	NLVEH	POINTS TO THE LAST ENTRY OF THE VEHICLE TABLE
/VEHS/	NOVEH	NUMBER OF ACTIVE VEHICLES IN FLEET. COMPUTED IN LEGPRO, USED IN SPRINT.
/VEHS/	NPREF	NUMBER OF VEHICLES INPUT TO VEHICLE PREFERENCE LIST
/VEHS/	NVEH	NUMBER OF VEHICLES IN THE VEHICLE TABLE
/VEHS/	NVHMAX	MAX. NO. OF VEHICLES ALLOWED IN THE VEHICLE TABLE.
		SET TO 30 IN DORCA
/VEHS/	NWVEH	NO. OF WORDS IN THE BASIC VEHICLE TABLE EQUAL TO 16
		EXCLUDING LEG DATA
/VEHS/	TBVEH	MATRIX CONTAINING 4 WORDS OF INFORMATION FOR EACH VEHICLE
		WORDS 1-2 VEHICLE NAME
		3 PACKED WORD CONTIANING START LOCATION AND LENGTH OF DATA FOR THIS VEHICLE IN THE DDB ARRAY.
		4 STAGES (UP TO 6 STAGES, PACKED 6 BITS/STAGE)
/VEHS/	VEH	VEH(I) = 1 IF VEHICLE I IS ACTIVE, = 0 IF NOT.
/VEHS/	VPREF	VPREF(N) IS A PACKED WORT CONTAINING INFORMATION ON THE N-TH VEHICLE IN THE VEHICLE PREFERENCE LIST
		BITS 0-17 VEHICLE INDEX IN TBVEH
		18-26 FIRST YEAR VEHICLE IS AVAILABLE (NORMALIZED TO RLDATE)
		27-36 LAST YEAR VEHICLE IS AVAILABLE
/YEAR/	FYEAR	FIRST YEAR THAT ANY CARGO IS SHIPPED, NORMALIZED TO RLDATE. E.G.- IF FYEAR = 1, THE JULIAN DATE IS RLDATE+1.
/YEAR/	IRELDT	THE YEAR TO WHICH ALL DATES ARE RELATIVE (I.E.-1970).
		SAME AS RLDATE BUT IN INTEGER FORMAT.
/YEAR/	LYEAR	LAST YEAR THAT ANY CARGO IS SHIPPED, NORMALIZED TO RLDATE. SEE FYEAR.
/YEAR/	NTYRS	ARRAY CONTAINING YEARS THAT CONTAINERS ARE USED
/YEAR/	NYRS	NUMBER OF YEARS SPANNED BY ALL MISSIONS.
/YEAR/	NYRS1	1 + NUMBER OF YEARS SPANNED BY ALL MISSIONS. NYRS1 = NYRS+1

/YEAR/	NYRS3	3 + NUMBER OF YEARS SPANNED BY ALL MISSIONS. NYRS3 = NYRS+3
/YEAR/	RLDATE	THE RELATIVE YEAR TO WHICH THE TIME IS BASED. I.E. 1970

APPENDIX B
MAJOR TABLES AND ARRAYS

This Appendix provides a general reference to the structure of the internal packed arrays and tables. The storage locations for all input data is described by the following table on page B-2. After the table appears a description of the format of individual tables with the routine that reads the cards and loads the table.

Storage of Input Data

Data	Where Stored	See subroutines
Container	Matrix TBCONT(I, J) Also DDB(K - NLCE) where K = ITBCNT*NWCE + NBCE	RDCONT, RDCRG
Leg	TBLEG(I, J)	RDLEG, RDVEH Also see below
Spread	TBSPD(I, J) Also DDB(NBSPD-NLSPD)	RDSPD
Vehicle	TBVEH(I, J) Also DDB(NBVEH-NLVEH)	RDVEH
Facility	DDB(NBFAC-NLFAC)	RDFAC
Cargo elements	DDB(NBCE-NLCE)	RDCRG, LEGPRØ, RDMISS
Program & mission	DDB(NBMISS-NLMISS) (Phase I cargo table) Also IVA and IFA arrays, PNAME and MNAME	RDMISS, LEGPRØ
Reports	IFLAG(1-12)	RDRPT

Leg Table

The leg data is read and stored in subroutine RDLEG and altered in subroutine RDVEH. The name and dimension of the leg table is TBLEG (12, 63), where 63 is the maximum number of legs and 12 is the number of words used for each leg. The contents of each 12-word group are:

Words	Contents
1 - 2	Leg name (A6, A4 format)
3 - 4	Name of next lower leg (A6, A4 format)
5	Maximum occupancy (deployment limit) - floating point
6	Packed word: (see below) where V_1, V_2, V_3, and V_4 are the indices of the default vehicles to be used on the leg and on the next lower, second lower, and third lower legs, respectively.
7	Index number of next lower leg (integer). Set to 63 if next lower leg is "NONE" (ie, if this leg originated on earth's surface).
8	Not used after subroutine RDVEH is finished.
9	Velocity increment ΔV (floating point).
10	Longshoring flag (floating point) (0-no, 1-yes).
11	Alternate vehicle index (integer).
12	Not used after RDVEH is finished.

Packed word (word 6):

0	12	18	24	30
0	V_1	V_2	V_3	V_4

The leg table has been sorted so that all legs with a common lower leg are grouped together.

Vehicle Table

Part of the vehicle data is stored in matrix TBVEH, which contains 5 words for each vehicle:

$name_1$	$name_2$	number	LOC	dry stages	wet stages

Words 1-2: Vehicle name (A6, A4) format

3 : Packed word
Bits 0-17 contain length N (number of cells) of data for this vehicle
Bits 18-35 contain location where data starts in DDB

4 : Packed word containing indices of up to 6 dry stages used by this vehicle, 6 bits/stage.

5 : Packed word containing indices of up to 6 wet stages used by this vehicle, 6 bits/stage.

The rest of the data is stored in DDB starting at cell LOC (see word 3 above) in 3 main sections: the basic set (16 words), option ISP data (10 cells), and one or more 5-word sets of leg information.

a) Basic Set (all floating point format except for name)

1	$Name_1$	$Name_2$	Propellant	Max flts/yr
5	Lifetime-fts	Lifetime-yrs	Min load	N. R. dev
9	Spread pointer	Prop tank	Rec. prod	Spread pointer
13	Deployment limit	Oper. cost	Refurb cost	Volume limit

Words 1-2 Name (A6, A4)
3 Propellant required
4 Maximum flights/year

5	Lifetime in flights
6	Lifetime in years
7	Minimum load (lbs)
8	Nonrecurring development cost
9	Pointer to spread table for development cost
10	Propellant tank index
11	Recurring production cost
12	Pointer to spread table for production cost
13	Deployment limit
14	Operations cost
15	Refurbishment cost
16	Volume capacity

b) ISP Data - Optional. May not be present.

1	"ISP"	ISP number	WSD	WNUP	WPMAX
5	WINT	WPBØ	WNIE	WACP	ISP number

Words	Contents
1	ISP (Hollerith)
2	ISP Number (specific Impulse)
3	WSD (dry structure weight)
4	WNUP (Non-usable propellant weight)
5	WPMAX (Maximum propellant weight)
6	WINT (Interstage weight)

7 WPBØ (Boil-off weight)

8 WNIE (Non-impulsive propellant weight)

9 WACP (Attitude control propellant weight)

10 ISP number (specific impulse)

c) Leg Data - At least one 5-word set

$Name_1$	$Name_2$	UPMAX	DNMAX	EXPMAX

Words 1-2 Leg name (A6, A4)

3 Maximum vehicle payload up (real)

4 Maximum payload down (real)

5 Maximum payload up if vehicle is expended (real)

Container Table

Container data is all stored in array TBCØNT, 8 words for each container.

$Name_1$	$Name_2$	Capacity	Empty wt
Classification	Volume	Temporary use	Return/Expend

Words	1-2	Container name (A6, A4)
	3	Capacity (lbs)
	4	Empty weight (lbs)
	5	Classification (1-crew capsule, 2-bulk container, 3-not used, 4-propellant tank)
	6	Volume
	7	Used for a flag in subroutine CNTRPT
	8	Return/expend option

Spread Table

Array TBSPD contains 3 words for each spread function input.

$Name_1$	$Name_2$	Pointer

Words 1-2 Function name (A6, A4)

3 Pointer to start of data for this function in DDB

The data itself consists of 2 + NYRS cells, NYRS being the number of years over which the cost is spread.

NYRS	YR1	F_1	F_2	F_3	Etc.

Word 1 NYRS - Number of years for spreading cost

2 YR1 - First year (e.g., 1975)

3, 4, 5 Etc. - F_i = factor for i^{th} year

Facility Table

Facility data is stored in DDB starting at location NBFAC in groups of 8 words/facility.

$Name_1$	$Name_2$	Life-yrs	Devel cost
Spread Pointer	-	Prod cost	Spread Pointer

Words 1-2	Facility name
3	Lifetime in years
4	Development cost
5	Spread function pointer for development cost
6	Not used
7	Production cost
8	Spread function pointer for production cost

Cargo Element Table

Cargo element data is stored in DDB starting at location NBCE in groups of 9 words/element.

$Name_1$	$Name_2$	$Description_1$
$Description_2$/IC1	$Description_3$/ICN	PNTR ¦ CLASS ¦ CAT
Up weight	Down weight	Volume

Words 1-2 Element name (A6, A4)

3 Description (A6)

Coupled items: contains coded word "COUPLE"

4 Description continued (A6)

Coupled items: first index IC1 in array CE in RDMISS (integer)

5 Description continued (A6)

Coupled items: last index ICN in array CE (integer)

6 Packed word

Bits 0-11: pointer to vehicle or facilities table

Bits 12-23: container class (1-crew, 2-bulk, 3-self contained discrete, 4-propellant, 5-coupled item)

Bits 24-35: category (1-material, 2-personnel, 3-facilities and satellites, 4-vehicles)

7 Up weight

8 Down weight

9 Volume

Contents of the DDB Array

Most of the input data and the big internally generated cargo tables are stored in the long DDB array. Various pointers are maintained to enable the program to find whatever information it wants. The order in which data is stored in DDB is:

Order	Type of Data	Subroutine Which Stores Data
1	Spread functions	RDSPD
2	Vehicle data	RDVEH
3	Facility data	RDFAC
4	Cargo elements	RDCRG
5	Cargo tables, phase I	LEGPRO, RDMISS
6	Excess core, if any	---
7	Cargo tables, phase II	LEGPRO

The phase II cargo tables arc stored top-down, starting at the high end of the DDB array and working backwards. This allows space for the phase I cargo tables to grow during the leg processing. If the data is so voluminous that the phase I and phase II overlap, the program will terminate itself.

For further details on the exact manner in which data is stored, see the writeups for the subroutines indicated.

Phase I Cargo Tables

Created initially by RDMISS to represent cargo shipments required by input mission data, the phase I cargo table is augmented as necessary by LEGPRO. The tables are sorted by vehicle, leg and year and processed in segments by the cargo assigner, each segment containing all cargo shipped on a given vehicle, leg and year. Each cargo item is represented by three adjacent words in the DDB array. The structure of these three packed words is as follows:

<table>
<tr><td></td><td>0</td><td></td><td></td><td></td><td></td><td>6</td><td>12</td><td></td><td></td><td>18</td><td>24</td><td>30</td></tr>
<tr><td>Word 1</td><td colspan="5">LEG</td><td>YR</td><td colspan="3">VEH</td><td>0</td><td>PROG</td><td>MISS</td></tr>
<tr><td>Word 2</td><td>SV</td><td>RT</td><td>DIR</td><td>DIS</td><td>PH</td><td>V2</td><td colspan="3">V3</td><td>V4</td><td colspan="2">CE #</td></tr>
<tr><td>Word 3</td><td>CP</td><td>PS</td><td colspan="5">CGN</td><td>SNG</td><td>0</td><td colspan="3">WT LF x 100000</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>14</td><td>15</td><td></td><td></td><td></td></tr>
</table>

Word	Bits	Contents
1	0-5	Leg number
	6-11	Year relative to 1970
	12-17	Vehicle index
	18-23	Zero (not used)
	24-29	Program number (index to PNAME table)
	30-35	Mission number (index to MNAME table)
2	0	Same vehicle. If flag = 1, this cargo item must make round trip on same vehicle flight. If flag = 0, item may make round trip on different flights.

	1	Round trip flag. If bit = 1, cargo item must travel both up and down. If bit = 0, item travels only one direction.
	2	Direction flag. If round trip flag is zero, this flag indicates direction (0 - up, 1 - down). If round trip flag is 1, this flag is irrelevant.
	3	Discrete flag. If flag = 1, this cargo item must now be treated as a discrete, indivisible item, regardless of specifications in cargo element table. If flag = 0, original specifications in cargo element table are to be used.
	4 - 5	Phase number of mission. 1 - initialization, 2 - sustaining, 3 - termination.
	6 - 11	Number of vehicle to be used on next lower leg, if any.
	12 - 17	Number of vehicle to be used on second lower leg, if any.
	18 - 23	Number of vehicle to be used on third lower leg, if any.
	24 - 35	Index number of this cargo item in cargo element table. Index number N means that this represents the N^{th} cargo element.
3	0	Coupled flag. If = 0, item is a normal single item. If = 1, item is all or part of a composite coupled item.

1	Primary/secondary flag. Not used for single items. For coupled items, 0 means it is the primary (whole composite) item, while 1 means it is one of the secondary (component) parts.
2-13	Composite group number (zero for non-composites). Used to tie together the primary and all secondaries.
14	Single deployment flag. If = 1, single deployment is required for this item; if = 0, multiple deployment is permitted.
15-17	Zero (not used)
18-35	Weight load factor for coupled items. For secondary items,

$$WTLF = \frac{\text{component weight}}{\text{total composite weight}} \times 100000$$

For primary items, WTLF ≡ 100000.
Integerized, not floating point.

Phase II cargo tables

These tables are created by LEGPRO from the phase I tables and the information provided by the cargo assigner (ASINER). Each cargo item is represented by 3 packed words in the DDB array:

LEG	VEH	YR	FLIGHT #		FLAGS
PROG	MISS	VEH	0	C. E. NUMBER	
LF x 100000			BLF x 100000		

Word	Bits	Contents
1	0 - 5	Leg number (index to leg table)
	6 - 11	Vehicle number (index to vehicle table)
	12 - 17	Year (relative date subtracted out)
	18 - 29	Flight number of this vehicle in this year on this leg.
	30	Same vehicle requirement. Now irrelevant.
	31	Round trip flag (see phase I). Now irrelevant.
	32	Direction flag (0 - up, 1 - down)
	33	Discrete flag (see phase I). Now irrelevant.
	34 - 35	Mission phase (1 - initialization, 2 - sustaining, 3 - termination)
2	0 - 5	Program number (index to program name table)
	6 - 11	Mission number (index to mission name table)
	12 - 17	Vehicle number (index to vehicle table) Same as bits 6 - 11 of word 1; used for sorting purposes.

Word	Bits	Contents
	18 - 23	Zero (not used)
	24 - 35	Cargo element number (index to cargo element table in DDB)
3	0 - 17	Load factor x 100000. (See below) Multiplied by 100000 and converted to an integer before being packed into this 18-bit portion of the word.
3	18 - 35	Bulk load factor x 100000. (See below) Multiplied by 100000 and converted to integer format before being packed into these 18 bits.

The "load factor" of any cargo item in the cargo table is defined as follows:

$$\text{load factor} = \left\{ \frac{\text{Item wt} \cdot \text{factor}}{\text{Tot wt up} + \text{tot wt down} \cdot \frac{(\text{UPMAX})}{(\text{DNMAX})}} \right.$$

where item wt = weight of the assigned cargo item (one direction only).
tot wt up = total weight carried upwards on this flight.
tot wt dn = total weight carried downwards on this flight.
UPMAX = maximum vehicle capacity (lbs.) upwards.
DNMAX = maximum vehicle capacity (lbs.) downwards.

$$\text{factor} = \begin{cases} 1 & \text{if cargo item is travelling upwards;} \\ \frac{\text{UPMAX}}{\text{DNMAX}} & \text{if cargo item is travelling downwards.} \end{cases}$$

For a single flight, the sum of the load factors for all cargo items travelling on that flight in both directions is 1. Note, however, that if UPMAX $\neq$ DNMAX, the load factor for an item travelling upwards will be different from the load factor for the same item travelling downwards.

The "bulk load factor" is of real interest only for bulk cargo which can be subdivided into several parcels packed into different containers; for non-divisible cargo items (crews and discretes), the bulk load factor must be 1. For bulk cargo, it is defined as follows:

$$\text{bulk load factor} = \frac{\text{assigned weight}}{\text{original weight}}$$

where assigned weight = weight of this portion of the bulk.

original weight = weight of this original whole bulk item input to the cargo element table.

Acquisition Tables

Mission input dictates how many of each kind of vehicle and facility must be shipped as part of the cargo. This information is used to initialize the vehicle and facility acquisition tables. During the leg processing, additional vehicles must be acquired to carry other cargo. The number of extra vehicles depends on the number of flights scheduled by the cargo assigner and the vehicles' lifetimes in years and number of flights. These extra vehicles are added to the vehicle acquisition table.

Vehicle acquisition table IVA

This array has NIVA entries. Each entry is one packed word having the following format:

Bits	Contents
0 - 5	Leg number
6 - 11	Vehicle number
12 - 23	Year
24 - 35	Number of units of this vehicle which must be acquired.

Facility acquisition table IFA

NIFA is the number of entries. Each entry is one packed word having the following format:

Bits	Contents
0 - 5	Program number
6 - 11	Mission number
12 - 23	Number of this facility as an entry in the cargo element table.
24 - 29	Year
30 - 35	Number of units of this facility to be acquired.

Flight Table

NFTBL is a long array describing all vehicle flights. Each word is a packed word containing 4 pieces of information

0	6	12	24
NEV	V #	YR	# of flights

Bits	0-5	Number of expended flights
	6-11	Index of vehicle
	12-23	Year relative to 1970
	24-35	Number of flights.

There is one such word generated for each call to ASINER from LEGPRO. If a given vehicle flies on more than one leg, there will be more than one word in NFTBL with the same vehicle index.

AEROSPACE CORPORATION

INTERNAL DISTRIBUTION LIST

(REFERENCE: COMPANY PRACTICE 7-21-1)

REPORT TITLE

DORCA II COMPUTER PROGRAM: Volume II Programmer's Manual

REPORT NO.	PUBLICATION DATE	SECURITY CLASSIFICATION
ATR-73(7315)-1, Vol. II	18 August 1972	Unclassified

(NOTE: FOR OFF-SITE PERSONNEL, SHOW LOCATION SYMBOL, e.g. JOHN Q. PUBLIC/VAFB)

J. H. Ashmore
E. Blond
N. R. Campbell
J. B. Carey
S. T. Chu
B. J. Gold (2)
A. Goldstein
G. M. Forslund
R. E. Kendall
E. I. Pritchard
J. Plough
T. Shiokari
L. R. Sitney
L. T. Stricker
S. M. Tennant
G. W. Timpson (3)
V. V. Voit
R. W. Wolfe (3)
S. T. Wray, Jr. (3)
L. Whittaker
J. K. Yakura
STSCF

APPROVED BY Robert R Wolfe DATE 9/12/72

SHEET ______ OF ______