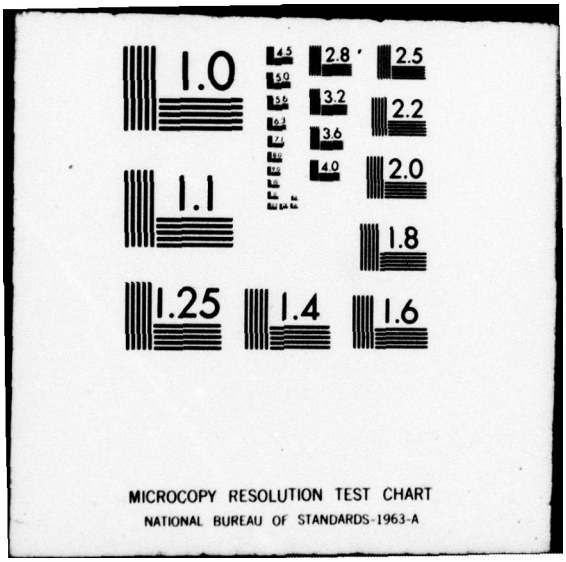
AD-A069 029 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CALIF F/G 9/2
LAMAS SYSTEM MANUAL.(U)
FEB 78 DAAG39-77-C-0112
UNCLASSIFIED NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DDC FILE COPY

LEVEL II

AD A069029

TRW
ADVANCED STUDIES
ONE SPACE PARK · REDONDO BEACH · CALIFORNIA 90278

LAMAS SYSTEM MANUAL

CDRL ITEM A008

FEBRUARY 1978

CONTRACT NO. DAAG39-77-C-0112

DDC
MAY 25 1979
C

This document has been approved for public release and sale; its distribution is unlimited.

Prepared for:
TRADOC Combined Arms Test Activity
U.S.A. Headquarters Center and School

DISCLAIMER NOTICE

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

TRW.
DEFENSE AND SPACE SYSTEMS GROUP
ADVANCED STUDIES
ONE SPACE PARK · REDONDO BEACH · CALIFORNIA 90278

(1)

(6)

LAMAS SYSTEM MANUAL

CDRL ITEM A008

(11) FEB[illegible] 1978

(12) 139p.

DDC
REPRODUCED
MAY 25 1979
C

(15) CONTRACT NO DAAG39-77-C-0112

241 60 70 62

Prepared for:
TRADOC Combined-Arms Test Activity
U.S.A. Intelligence Center and School

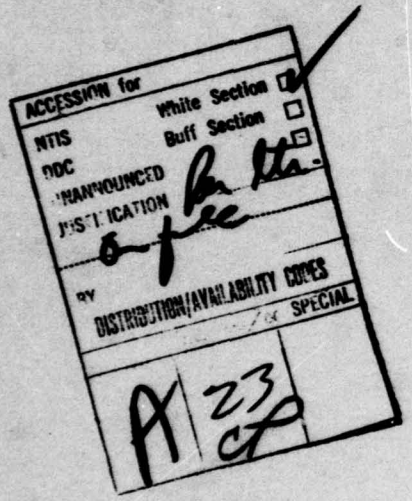
This document has been approved for public release and sale; its distribution is unlimited.

409637

1B

LAMAS SYSTEM MANUAL

This document describes the LAMAS software package from both a programmer and user standpoint. The text is written in Program Design Language (PDL) format and represents the final version of the working document used throughout program development. All LAMAS software and manual procedures are described, with the exception of the display software developed using company funds. A proprietary version of this document which describes both deliverable and company-funded software is maintained on file in the TRW Advanced Studies office.



TRW SYSTEMS GROUP
ONE SPACE PARK
REDONDO BEACH, CALIFORNIA 90278
(213) 535-4321

LOCATION AND MOVEMENT ANALYSIS SYSTEM

(LAMAS)

SOFTWARE TO COMPUTE GROUND FORCE MOVEMENT CONSIDERING MOVEMENT DOCTRINE, ROAD CONDITIONS, AND RISK FACTORS

21 FEB 78

PDL 03.06

TABLE OF CONTENTS

GENERAL DESCRIPTION 2
OVERVIEW 3
INITIALIZATION 4
GROUND MOVEMENT CALCULATIONS 5
USER FORMAT 6
SYSTEM ARCHITECTURE 7

DATA BASE DESCRIPTIONS 8
NODES AND LINKS 9
NODE VECTOR FORMAT 10
NODE VECTOR DESCRIPTION 11
TERRAIN 12
CROSS-COUNTRY AND CONCEALMENT CODE DESCRIPTIONS 13
UNITS 14
UNIT-VECTOR FORMAT 15
UNIT-VECTOR DESCRIPTION 16
SOLUTIONS 17
SOLUTION-VECTOR FORMAT 18
SOLUTION-VECTOR DESCRIPTION 19
ROUTES 20
ROUTE-VECTOR FORMAT 21
ROUTE-VECTOR FORMAT DESCRIPTION 22
MAP DIRECTORY 23
CROSS-COUNTRY DIRECTORY 24

PREPARATION FOR USE 25
PREPARATION 26
CREATE NODE-VECTOR AND MAP-DIRECTORY FILES ON DISK 27
CREATE NODE CARD 28
NODE CARD FORMAT 29
CREATE LINK CARD 30
LINK CARD FORMAT 31
CREATE MAP-DIRECTORY ENTRY FOR THIS MAP 32
CREATE CROSS COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK 33

LAMAS FLOW 34

TABLE OF CONTENTS

GENERAL DESCRIPTION 2
OVERVIEW 3
INITIALIZATION 4
GROUND MOVEMENT CALCULATIONS 5
USER FORMAT 6
SYSTEM ARCHITECTURE 7

DATA BASE DESCRIPTIONS 8
NODES AND LINKS 9
NODE VECTOR FORMAT 10
NODE VECTOR DESCRIPTION 11
TERRAIN 12
CROSS-COUNTRY AND CONCEALMENT CODE DESCRIPTIONS 13
UNITS 14
UNIT-VECTOR FORMAT 15
UNIT-VECTOR DESCRIPTION 16
SOLUTIONS 17
SOLUTION-VECTOR FORMAT 18
SOLUTION-VECTOR DESCRIPTION 19
ROUTES 20
ROUTE-VECTOR FORMAT 21
ROUTE-VECTOR FORMAT DESCRIPTION 22
MAP DIRECTORY 23
CROSS-COUNTRY DIRECTORY 24

PREPARATION FOR USE 25
PREPARATION 26
CREATE NODE-VECTOR AND MAP-DIRECTORY FILES ON DISK 27
CREATE NODE CARD 28
NODE CARD FORMAT 29
CREATE LINK CARD 30
LINK CARD FORMAT 31
CREATE MAP-DIRECTORY ENTRY FOR THIS MAP 32
CREATE CROSS COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK 33

LAMAS FLOW 34

LAMAS 35
INPLEMENT ALGORITHMS AND FUNCTIONS USING THE NODE←VECTUR DATA BASE 36
PREPARE FUR ROAD NETWORK PATH CALCULATIONS 37
INITIALIZE MAP←NUMBERS AND SCREEN EXTREMES 38
ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS 39
READ THIS MAP'S NODES INTO MAIN MEMORY 40
CHANGE ADJACENT NODE'S MAP←NUMBER TO INDEX 41
INITIALIZE UNIT←VECTORS 42
ESTABLISH A FORWARD EDGE OF BATTLE AREA 43
FIND NODE←VECTOR NEAREST TO UTM COORDINATES 44
PATH DETERMINATION AND DISPLAY 45
PERFORM INTERDICTIVE OPERATIONS 46
PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINATES 47
CHANGE CONTENTS OF A NODE←VECTOR 48
CHANGE CHARACTERISTIC 49
DELETE A NODE FROM THE NODE←VECTOR ARRAY 50
PRINT CONTENTS OF A NODE←VECTOR 51
CHANGE PROPERTIES OF A UNIT 52
CALCULATE PATH 53
PURGE ALL ROUTES 54
REINITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES 55
CALCULATE AND ASSIGN TIME AND WORTH VALUES 56
COMPUTE NEXT NODE TO LABEL 57
EXPLANATION OF PATH DETERMINATION ALGORITHM 58
COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR 59
CALCULATE SECOND BEST PATH 60
FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL 61
CALCULATE BEST NODE AT WHICH TO INTERDICT 62
PRESENT RESULTS 63
PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES 64
PRINT PATH STATISTICS 65
IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE CROSS-COUNTRY DATA BASE 66
PERFORM TERRAIN INITIALIZATION 67
MAP←NUMBER TO BE USED 68
ESTABLISH UNIT←VECTORS 69
CHANGE ENTERED COORDINATES TO AN INDEX INTO THE DATA BASE 70
PERFORM TERRAIN PATH CALCULATIONS 71
CALCULATE CROSS-COUNTRY PATH 72
OBTAIN UNIT NAMES TO BE CONSIDERED 73
OBTAIN THE PATH PARAMETERS 74
CALCULATE THE CRUSS-COUNTRY PATH 75

CALCULATE THE ADJACENT NODE'S TIME VALUES 76
CONSTRUCT SOLUTION←VECTORS AND ROUTE←VECTOR 77
DISPLAY RESULTS 78

DATA INDEX 79

FLOW SEGMENT INDEX 80

SEGMENT REFERENCE TREES 81

GENERAL DESCRIPTION

OVERVIEW

```
#                                                                                #
#  1     THE PURPOSE OF THIS SOFTWARE PACKAGE IS TO COMPUTE GROUND                #      10
#  2    FORCE MOVEMENT USING OPTIMAL PATH DETERMINATION AND ROUTE CONFLICT        #      11
#  3    RESOLUTION ALGORITHMS.  TO INSURE FLEXIBILITY, USER INTERACTION IS        #      12
#  4    SUPPORTED DURING MOST PHASES OF THE SOFTWARE'S OPERATION.                 #      13
#  5     THREE MAJOR AREAS ARE INVOLVED IN THE PROCESS; INITIALIZATION           #      14
#  6    GROUND FORCE MOVEMENT CALCULATIONS, AND DISPLAY.                          #      15
#                                                                                #
```

INITIALIZATION

```
#                                                                                #
#  1    INITIALIZATION IS THE PROCESS OF PREPARING DATA BASES FOR                #  17
#  2   GENERAL USE.  THERE ARE THREE BASIC TYPES OF DATA BASES; PERMA-           #  18
#  3   NENT, TEMPORARY - USER DEFINED, AND TEMPORARY - PROGRAM DEFINED.          #  19
#  4    PERMANENT DATA BASES ARE CREATED BEFORE THE USER EVER                    #  20
#  5   INVOKES THE SYSTEM, AND ARE GENERALLY UNALTERABLE BY THE USER.           #  21
#  6   INCLUDED ARE SUCH FILES AS NODES, ROADS, CITIES, CROSS-COUNTRY MOVE-     #  22
#  7   MENT, CONCEALMENT, MAP←DIRECTORY, ROAD←DIRECTORY, AND CROSS-COUNTRY      #  23
#  8   DIRECTORY.                                                               #  24
#  9    A USER-DEFINED TEMPORARY DATA BASE IS A FILE OVER WHICH THE             #  25
# 10   USER HAS CONSIDERABLE CONTROL, BUT WHICH EXISTS ONLY FOR A PAR-         #  26
# 11   TICULAR SESSION ON THE SYSTEM.  ONCE THE USER SIGNS OFF, THE DATA       #  27
# 12   BASE IS LOST.  THE UNITS DATA BASE IS THE PRIMARY EXAMPLE OF THIS       #  28
# 13   TYPE IN LAMAS.  HERE THE USER DEFINES THE CONTENTS, AND MAY ALTER       #  29
# 14   THE CONTENTS AT ANY TIME, BUT THE FILE DOES NOT CARRY OVER FROM         #  30
# 15   ONE SESSION TO ANOTHER.                                                 #  31
# 16    PROGRAM-DEFINED TEMPORARY DATA BASES ARE FILES OVER WHICH              #  32
# 17   THE USER HAS NO CONTROL WHATSOEVER.  THE FILES MAY ENABLE THE USER      #  33
# 18   TO DISPLAY SOME RESULTS, BUT NO ALTERATIONS MAY BE MADE.  EX-           #  34
# 19   AMPLES OF THIS TYPE OF DATA BASE ARE SOLUTION←VECTORS, ROUTE←           #  35
# 20   VECTORS, MAP←DIRECTORY IN MEMORY, AND ROAD←DIRECTORY IN MEMORY.         #  36
# 21    THE EFFECT OF INITIALIZATION VARIES FROM FILE TO FILE.                 #  37
# 22   FOR SOME, IT IS JUST SETTING ALL ELEMENTS TO ZERO, AS IN THE CASE       #  38
# 23   OF THE SOLUTION←VECTORS AND ROUTE←VECTORS.  FOR OTHERS, IT IS THE       #  39
# 24   ESTABLISHING OF VALUES, AS IN ALL OF THE PERMANENT FILES, UNITS DATA     #  40
# 25   BASE, AND THE DIRECTORIES.                                              #  41
# 26    THE MANNER IN WHICH THE VALUES ARE SET ALSO VARIES FROM                #  42
# 27   FILE TO FILE.  ALL PERMANENT FILES ARE CREATED BY TRANSFERRING          #  43
# 28   INFORMATION FROM DATA CARDS TO DISK STORAGE.  THE UNITS FILE IS         #  44
# 29   ESTABLISHED BY DIRECT USER INPUT, WHILE THE NODE←VECTOR FILE            #  45
# 30   IN MEMORY AND THE DIRECTORIES IN MEMORY ARE CREATED BY THE PRO-         #  46
# 31   GRAM, BUT ONLY AFTER THE USER HAS INVOKED PARTICULAR ROUTINES.          #  47
#                                                                              #
################################################################################
```

GROUND MOVEMENT CALCULATIONS

```
#                                                                                #
#  1    THERE ARE TWO BASIC TYPES OF GROUND MOVEMENT CALCU-                      #      49
#  2   LATIONS WE WILL MAKE IN THIS PACKAGE.  ONE CONSIDERS THE                  #      50
#  3   ROAD NETWORK FOR THE AREA OF INTEREST AS REPRESENTED IN THE               #      51
#  4   NODE-VECTOR DATA BASE, WHILE THE OTHER USES CROSS-COUNTRY                 #      52
#  5   MOVEMENT INFORMATION AS SUPPLIED BY THE ARMY ON ITS COM AND               #      53
#  6   CONCEALMENT MAPS FOR THE AREA OF INTEREST.                               #      54
#  7    THE MAJOR DIFFERENCES BETWEEN THESE TWO APPROACHES                      #      55
#  8   ARE IN THEIR SCOPE OF APPLICATION, AND THE DATA BASES WHICH              #      56
#  9   SUPPLY THE INFORMATION.  USING THE ROAD NETWORK APPROACH, ONE            #      57
# 10   MAY CONSIDER UP TO TEN 1:50000 MAPS FOR ANY COMPUTATION; WHILE          #      58
# 11   THE CROSS-COUNTRY MOVEMENT MUST BE LIMITED TO AT MOST ONE               #      59
# 12   1:50000 MAP.  THIS IS DUE TO THE MANNER IN WHICH THE TWO TYPES          #      60
# 13   ARE MODELED.                                                            #      61
# 14    BECAUSE OF THE DIFFERENCE IN INFORMATION BEING HELD IN                 #      62
# 15   THE DATA BASE REGARDING THE TWO TYPES OF MOVEMENT, THIS CALLS           #      63
# 16   FOR TWO VERY DIFFERENT DATA BASE APPROACHES.  THESE WILL BE             #      64
# 17   EXPLAINED LATER ON IN THIS DOCUMENT.                                    #      65
# 18    ON THE OTHER HAND, THE ALGORITHM USED TO CALCULATE THE                 #      66
# 19   PATHS FOR EACH TYPE OF MOVEMENT IS VIRTUALLY IDENTICAL.  THE            #      67
# 20   ONLY DIFFERENCE LIES IN THE INTERPRETATION OF THE DATA BASES;           #      68
# 21   THE ACTUAL MECHANICS OF THE PATH CALCULATIONS ARE THE SAME.             #      69
# 22    THE ALGORITHM FINDS THE 'BEST' PATH FOR A UNIT TO TRAVEL.              #      70
# 23   'BEST' IS A USER-DEFINED PARAMETER MADE UP OF TWO COMPONENTS,           #      71
# 24   RISK AND TIME.  RISK IS A VALUE ARRIVED AT BY CONSIDERING               #      72
# 25   ROAD CONDITIONS, TERRAIN FACTORS, WHETHER OR NOT A BRIDGE IS            #      73
# 26   PRESENT, AND WHETHER OR NOT A CITY IS PRESENT.  TIME IS THE             #      74
# 27   AMOUNT OF TIME IT TAKES A PARTICULAR UNIT TO TRAVEL FROM ONE            #      75
# 28   NODE TO ANOTHER.                                                        #      76
# 29    OTHER TYPES OF PATH CALCULATIONS MAY BE PERFORMED AS                   #      77
# 30   WELL.  THE USER MAY FIND THE SECOND BEST PATH TO TRY TO ATTACK          #      78
# 31   THE SENSOR TASKING PROBLEM, HE MAY FIND THE BEST NODE AT WHICH          #      79
# 32   TO PERFORM INTERDICTION FOR DISRUPTION PURPOSES, OR HE MAY FIND         #      80
# 33   THE BEST LONGEST PATH (A METHOD OF CALCULATING HOW QUICKLY A FORCE      #      81
# 34   MAY BUILD UP IN A PARTICULAR AREA.                                      #      82
#                                                                                #
##################################################################################
```

USER FORMAT

```
#                                                                                #
#  1        THE FORMAT FOR USER INTERACTION WILL BE OF A SIMPLE NATURE.          #      84
#  2    WHEN STARTING, A BASIC MENU OF AVAILABLE FUNCTIONS WILL BE LISTED ON     #      85
#  3    A DISPLAY SCREEN.  THE USER WILL DECIDE WHICH ONE HE WISHES TO PURSUE,   #      86
#  4    AND ENTER ITS CODE AT A TERMINAL.  THIS MAY CAUSE DIRECT ACTION, OR      #      87
#  5    CAUSE ANOTHER MENU TO BE DISPLAYED.  THROUGHOUT THE OPERATION, WHETHER   #      88
#  6    OR NOT A MENU APPEARS, THE USER WILL BE PROMPTED WITH INSTRUCTIONS       #      89
#  7    REGARDING HOW TO PROCEED.  IN THIS WAY, THE USER HAS 'HANDS-ON' CONTROL  #      90
#  8    OF ANY AND ALL ACTIVITIES INVOLVED IN THE GROUND FORCE MOVEMENT COMPU-   #      91
#  9    TATIONS.                                                                 #      92
#                                                                                #
```

SYSTEM ARCHITECTURE

THIS SOFTWARE HAS BEEN DESIGNED TO OPERATE ON A PDP 11/45 COMPUTER HAVING 96K WORDS OF MEMORY, WITH ANY SINGLE TASK ABLE TO ACCESS AT MOST 32K WORDS. THE OPERATING SYSTEM, RSX-11M, SUPPORTS A USER DEFINED MULTITASKING CAPABILITY, WITH THE TOTAL AMOUNT OF MEMORY USAGE TO NOT EXCEED 64K WORDS.

FOUR DIFFERENT PERIPHERALS ARE NEEDED TO EFFECTIVELY USE THIS SOFTWARE PACKAGE. ONE IS A TERMINAL FOR COMMUNICATION WITH THE PROGRAM. THIS MAY BE A DECWRITER, A TEKTRONIX CRT, OR A VT05 TERMINAL. ALL PERMANENT DATA BASES ARE STORED ON AN RP04 MOVING HEAD DISK UNIT, AS WELL AS ALL OF THE SOURCE AND OBJECT CODE, AND TASK IMAGE. A GOULD 4800 HIGH-SPEED LINE PRINTER IS USED FOR PATH STATISTICS OUPUT, AND A COMTAL 8500 IMAGE PROCESSING SYSTEM IS USED FOR THE VISUAL DISPLAYS.

DUE TO MEMORY CONSIDERATIONS, SUCH AS THE LARGE AMOUNT OF SPACE NEEDED FOR DATA BASES, IT HAS BEEN DECIDED TO USE MULTIPLE TASKING TO IMPLEMENT THIS PROGRAM. ONE TASK PERFORMS ALL OF THE I/O FUNCTIONS, THE OBTAINING OF USER INPUT AND THE PRINTING OF MESSAGES, WHILE ANOTHER DOES ALL OF THE COMPUTATION. RSX-11M IS STRUCTURED SO THAT THIS MAY BE DONE WITH A MINIMUM OF DELAY TIME USED TO SWITCH FROM ONE TASK TO THE OTHER. MESSAGES MAY BE SENT BETWEEN THE TWO, AND THIS IS HOW OUTPUT DATA IS TRANSFERRED. THERE IS ALSO A SEPARATE GRAPHICS PACKAGE WHICH IS ITSELF ANOTHER TASK, SO THERE ARE THREE TASKS RUNNING CONCURRENTLY IN THIS SYSTEM.

THE TOTAL MEMORY OF THESE THREE TASKS MUST NOT EXCEED 64K WORDS, OR SEVERE DEGRADATION OF RESPONSE TIME IS INCURRED. THIS NECESSITATES THE USE OF OVERLAYS IN ORDER TO SAVE MEMORY SPACE.

WE HAVE ARRIVED AT THESE ESTIMATES FOR STORAGE REQUIREMENTS:

DESCRIPTION	CONTENTS	MEMORY USED
IN←CORE←NODE←ARRAY	400 NODES AT 32 WORDS EACH (ABOUT 10 MAPS)	12,800 WORDS
SOLUTION←VECTORS	40 ROUTES WITH 25 NODES PER ROUTE.	5,000 WORDS

```
# 37   OTHER                    DIRECTORIES, COUNTERS, ETC.        5,000 WORDS          #   130
# 38   .                                                                                 #   131
# 39   TOTAL                                                      22,800 WORDS          #   132
# 40   .                                                                                 #   133
# 41   .                                                                                 #   134
# 42        THUS APPARENTLY LEAVING US WITH 10,000 WORDS FOR ACTUAL EXECUTABLE          #   135
# 43        CODE PER OVERLAY.  HOWEVER, THE GRAPHICS PACKAGE USES 21,300 WORDS,         #   136
# 44        AND THE I/O ROUTINES USE APPROXIMATELY 17,000 WORDS, SO ONLY ABOUT          #   137
# 45        4,000 WORDS ARE TRULY AVAILABLE FOR CODING IN THE MAIN ROUTINE.             #   138
# 46   .                                                                                 #   139
# 47   TIMINGS:     TO BE DETERMINED                                                    #   140
#                                                                                       #
########################################################################################
```

DATA BASE DESCRIPTIONS

NODES AND LINKS

CONSISTING OF ONLY NODE-VECTORS. THIS DATA BASE IS THE LARGEST, AND PROBABLY THE MOST IMPORTANT, OF ALL THE DATA BASES. THE INFORMATION CONTAINED WITHIN ENABLES THE PATH ALGORITHM TO ACCURATELY CALCULATE THE BEST PATH FOR A GIVEN UNIT TO TRAVEL.

A NODE-VECTOR IS A 32-WORD STORAGE AREA WHICH CONTAINS DATA PERTAINING TO A NODE AND THE LINKS WHICH CONNECT IT TO OTHER NODES. A NODE IS A POINT OF INTEREST, USUALLY, BUT NOT NECESSARILY, BEING AT THE JUNCTION OF TWO OR MORE ROADS. THE LINK BETWEEN TWO NODES IS MOST OFTEN A REPRESENTATION OF THE ROADWAY JOINING THE TWO, BUT IT CAN ALSO BE A REPRESENTATION OF SIMPLY THE GENERAL PATH (CROSS-COUNTRY OR OTHERWISE) WHICH CONNECTS THEM. TWO NODES ARE SAID TO BE ADJACENT TO EACH OTHER IF THERE EXISTS A SINGLE LINK BETWEEN THEM. A PATH IS THEN A SERIES OF LINKS SUCH THAT THE LINKS, WHEN TAKEN AS A WHOLE, ARE CONNECTED. THAT IS, STARTING AT NODE 1, A LINK CONNECTS WITH NODE 2 ADJACENT TO NODE 1, WHICH CONNECTS WITH NODE 3 ADJACENT TO NODE 2, ETC., UNTIL THE DESTINATION NODE IS REACHED. ALL NODE-VECTOR DATA HAS BEEN OBTAINED FROM STANDARD L SERIES 1:50000 MAPS.

A GROUND FORCE UNIT'S MOVEMENT THROUGH A ROAD NETWORK IS CALCULATED BY MODELING ITS TRAVEL FROM ONE NODE TO ANOTHER ALONG A LINK. A LINK IS A CHARACTERIZATION OF A PATH, AND AS SUCH, PASSES ALONG ASSOCIATED PIECES OF INFORMATION, SUCH AS: DISTANCE, NUMBER OF LANES, ROAD TYPE, OFF-ROAD TRAFFICABILITY, AND TERRAIN, BRIDGE, AND CITY CODES.

DISTANCE REFERS TO THE ACTUAL DISTANCE IN KILOMETERS ONE WOULD HAVE TO TRAVEL TO MOVE FROM ONE NODE TO ANOTHER ALONG THE TRUE CONNECTION. NUMBER OF LANES MEANS THE NUMBER OF SINGLE-FILE ROAD LANES WHICH ARE EQUIVALENT TO THE LINKING ROAD. ROAD TYPE IS A CODE REFLECTING ANY OF FOUR POSSIBILITIES: AUTOBAHN OR AUTOSTRADE, MAIN ROAD, SECONDARY ROAD, OR FAIR WEATHER ROAD ONLY. OFF-ROAD TRAFFICABILITY IS ANOTHER CODE, THIS TIME INDICATING OPEN TRAFFICABLE AREAS NEAR THE ROAD WHICH WILL ALLOW REST OR ALTERNATE ROUTES. TERRAIN MEASURES LAND FEATURES AND INTERVISIBILITY. THE CHOICES ARE FLAT, HILLY, MOUNTAINOUS, OR ANY OF THESE THREE PLUS HIGH TERRAIN IMMEDIATELY TO THE WEST. THE BRIDGE CODE INDICATES THE PRESENCE, OR LACK OF, A BRIDGE, AND IF IT EXISTS, ITS SIZE, EITHER SMALL, MEDIUM, OR LARGE. FINALLY, CITY CODE FUNCTIONS IN A MANNER IDENTICAL TO THE BRIDGE CODE. RISK IS TAKEN TO BE A LINEAR COMBINATION OF OFF-ROAD TRAFFICABILITY, TERRAIN, BRIDGE,

AND CITY CODES.

THE CROSS-COUNTRY AND CONCEALMENT PATH ALGORITHM DOES NOT USE THE NODES AND LINKS DATA BASE.

THE ELEMENTS WHICH CHARACTERIZE A NODE ARE SIGNIFICANTLY DIFFERENT THAN THOSE CHARACTERIZING A LINK. ALL NODAL INFORMATION IS LOCATION ORIENTED, THAT IS, IT DESCRIBES EXACTLY WHERE THE NODE IS ON THE MAP. THIS INFORMATION INCLUDES LATITUDE, LONGITUDE, THE MAP NUMBER FOR THE PARTICULAR MAP ON WHICH THE NODE RESIDES, AND THE NODE NUMBER (AN ARBITRARY FIGURE ESTABLISHED DURING THE MANUAL CREATION OF THE DATA BASE.)

A NODE←VECTOR THEN, IS COMPRISED OF THE ABOVEMENTIONED NODAL AND LINK INFORMATION (A MAXIMUM OF FOUR ADJACENT NODES MAY EXIST), PLUS SEVERAL OTHER VALUES, WHOSE SIGNIFICANCE WILL BECOME CLEAR LATER ON IN THIS DOCUMENT (AMONG THESE ARE WORTH MEASURE, PREDECESSOR NODE, CUMMULATIVE TIME, CUMMULATIVE WORTH MEASURE, TIME MEASURE, SOLUTION VECTOR POINTER, LANE USED, RISK MEASURE, AND PARK TIME. THESE ARE ALL USED BY THE PATH ALGORITHM.)

NODE←VECTORS MAKE UP A PERMANENT DATA BASE WHICH RESIDES ON DISK. ONE MAY THINK OF THIS STORAGE AS BEING DIVIDED INTO BINS; SUCH THAT ONE BIN CONTAINS THE NODE←VECTORS FOR ONE MAP. A DIRECTORY IS ALSO STORED ON DISK, AND IT CONTAINS IDENTIFICATION FOR EACH BIN; THAT IS, THE APPROPRIATE MAP NUMBER. ACCESS OF ANY MAP'S NODE←VECTORS IS QUITE SIMPLE; ONE FINDS THE MAP NUMBER IN THE DIRECTORY. THE DIRECTORY ENTRY POINTS TO THE BIN ON DISK WHICH CONTAINS THE NODE←VECTORS, WHICH MAY THEN BE READ INTO MAIN MEMORY.

ONCE IN MAIN MEMORY, NODE←VECTORS ARE ALTERABLE BY THE USER. FUNCTIONS EXIST WHICH ALLOW THE USER TO CHANGE CERTAIN VALUES; SUCH AS LINK DISTANCE, ROAD TYPE, OR NUMBER OF LANES. ANY CHANGES MADE ARE NOT PERMANENT; SINCE THE "CORRECTED" VERSION OF THE DATA BASE IS NEVER COPIED ONTO THE PERMANENT DISK FILE.

WHILE IN MEMORY, THE NODE←VECTORS ARE THOUGHT OF AS OCCUPYING A NODE←VECTOR ARRAY. MAIN MEMORY IS LINEARLY ORGANIZED, SO IT IS NATURAL TO THINK OF NODE←VECTORS COMPRISING A LIST INDEXABLE BY INTEGERS. THUS, IF TWO MAPS' NODE←VECTORS ARE IN MEMORY, WITH THE FIRST MAP HAVING 30 NODE←VECTORS, THE FIRST NODE←VECTOR OF THE SECOND MAP IS THE 31 ST NODE←VECTOR IN THE ARRAY, ETC.

NODE VECTOR FORMAT

	WORD #	BYTE 1	BYTE 2
1		NOTE THAT EACH WORD IS SIXTEEN BITS,	
2		AND IT IS OFTEN POSSIBLE TO CONSIDER TWO	
3		PIECES OF INFORMATION FOR ONE WORD, WITH	
4		EACH PIECE OCCUPYING ONE BYTE. IF THERE	
5		IS NO SECOND INFORMATION PIECE IN A WORD,	
6		THE DATA IS RIGHT-ADJUSTED WITHIN THE WORD.	
7			
8	1	NODE MAP+NUMBER	
9	2	NODE # WITHIN MAP	$LABEL
10	3	WORTH MEASURE	
11	4	PREDECESSOR NODE NUMBER	
12	5	CUMMULATIVE TIME	
13	6	CUMMULATIVE WORTH MEASURE	
14	7	TIME MEASURE	
15	8	SOLUTION-VECTOR POINTER	
16	9	LATITUDE	
17	10	LONGITUDE	
18	11	NUMBER OF ADJACENT NODES	$LANE USED
19	12	ADJACENT MAP #1	
20	13	NODE+NUMBER WITHIN MAP	$LINK DISTANCE
21	14	# OF LANES	$ROAD TYPE
22	15	OFF-ROAD TRAFFICABILITY	$TERRAIN
23	16	BRIDGE	$CITY
24	17	ADJACENT MAP # 2	
25	18	NODE # WITHIN MAP	$LINK DISTANCE
26	19	# OF LANES	$ROAD TYPE
27	20	OFF-ROAD TRAFFICABILITY	$TERRAIN
28	21	BRIDGE	$CITY
29	22	ADJACENT MAP #3	
30	23	NODE # WITHIN MAP	$LINK DISTANCE
31	24	# OF LANES	$ROAD TYPE
32	25	OFF-ROAD TRAFFICABILITY	$TERRAIN
33	26	BRIDGE	$CITY
34	27	ADJACENT MAP #4	
35	28	NODE # WITHIN MAP	$LINK DISTANCE
36			

# 37	29	# OF LANES	$ROAD TYPE	#	251
# 38	30	OFF-ROAD TRAFFICABILITY	$TERRAIN	#	252
# 39	31	BRIDGE	$CITY	#	253
# 40	32	RISK MEASURE	$PARK TIME	#	254
#				#	

##########

NODE VECTOR DESCRIPTION

WORD #	DESCRIPTION AND USE	RANGE
1	NUMBER OF 1:50000 MAP FROM WHICH DATA IS TAKEN.	1-32767
2,1	NUMBER OF NODE WITHIN THE MAP. THESE NUMBERS ARE SELECTED WHEN THE NODES ARE MANUALLY DEFINED.	1-80
2,2	FLAG TO INDICATE WHETHER OR NOT THIS NODE HAS BEEN LABELED BY THE PATH ALGORITHM.	0-1
3	WORTH OF THE LINK BETWEEN THIS NODE AND ITS PREDECESSOR, AS DETERMINED BY THE PATH ALGORITHM.	0-100
4	NODE+VECTOR INDEX OF THE PREDECESSOR NODE+VECTOR, SELECTED BY THE OPTIMAL MOVE ALGORITHM.	1-400
5	THE CUMULATIVE TIME, IN MINUTES, OF A MOVEMENT UNIT AT THIS NODE POSITION AS DETERMINED BY THE OPTIMAL MOVEMENT ALGORITHM. THERE IS A MAXIMUM TRAVEL TIME OF TEN DAYS (=14400 MINUTES) PLUS 23 HOURS, 59 MINUTES (=1439 MINUTES).	0-15839
6	TOTAL WORTH MEASURE OF THE PATH FROM INITIAL NODE UP TO AND INCLUDING THIS LINK'S WORTH MEASURE (WORD 3).	0-32767
7	THE ACTUAL TIME FOR A UNIT TO PROGRESS FROM PREDECESSOR TO THIS NODE.	0-15839
8	INDEX TO ENTRY IN THE SOLUTION+VECTOR ARRAY. THIS POINTER ENABLES THE NODE+VECTOR TO BE ASSOCIATED WITH THE OPTIMAL ROUTE DATA. ONLY NODES THAT ARE PART OF AN OPTIMAL SOLUTION WILL HAVE THIS ENTRY. IF THE NODE IS PART OF SEVERAL OPTIMAL ROUTES, THIS WORD WILL POINT TO THE	0-5000

# 37		MOST RECENTLY CALCULATED SOLUTION←VECTOR.		#	292
# 38				#	293
# 39	9	NODE LATITUDE IN KILOMETERS FROM EQUATOR.	0-32767	#	294
# 40		THIS DATA IS ONLY USED FOR DISPLAY AND IS THUS		#	295
# 41		+ OR - 1 KM RESOLUTION. ROUTE ALGORITHMS USE LINK		#	296
# 42		DISTANCES, NOT THESE LATITUDE, LONGITUDE COOR-		#	297
# 43		DINATES.		#	298
# 44				#	299
# 45	10	NODE LONGITUDE IN KILOMETERS FROM GREENWICH.	0-32767	#	300
# 46		THIS DATA IS ONLY USED FOR DISPLAY AND IS THUS		#	301
# 47		+ OR - 1 KM RESOLUTION.		#	302
# 48				#	303
# 49	11,1	NUMBER OF ADJACENT NODES, MAXIMUM IS FOUR,	1-4	#	304
# 50		MINIMUM IS ONE.		#	305
# 51				#	306
# 52	11,2	LANE NUMBER USED DURING PATH CALCULATION. THIS	1-4	#	307
# 53		IS NECESSARY IN ORDER TO EFFECTIVELY USE MULTIPLE		#	308
# 54		LANED ROADS.		#	309
# 55				#	310
# 56	12	THE MAP←NUMBER OF THE FIRST ADJACENT NODE.	1-32767	#	311
# 57		THIS ENTRY WILL BE RESET AFTER DATA IS READ		#	312
# 58		FROM DISK TO CORE TO BECOME THE NODE←VECTOR		#	313
# 59		ARRAY INDEX OF THE FIRST ADJACENT NODE.		#	314
# 60				#	315
# 61	13,1	NUMBER OF NODE WITHIN THE MAP OF THE FIRST	1-60	#	316
# 62		ADJACENT NODE.		#	317
# 63				#	318
# 64	13,2	LINK DISTANCE IN TENTHS OF A KM BETWEEN THE	0-255	#	319
# 65		NODE AND THE FIRST ADJACENT NODE.		#	320
# 66				#	321
# 67	14,1	NUMBER OF SINGLE FILE ROAD LANES BETWEEN	1-4	#	322
# 68		THE NODE AND THE FIRST ADJACENT NODE.		#	323
# 69				#	324
# 70	14,2	ROAD TYPE CODE. 1=AUTOBAHN OR AUTOSTRASE,	1-4	#	325
# 71		2=MAIN ROAD, 3=SECONDARY ROAD OR ROAD, 4=		#	326
# 72		FAIR WEATHER ROAD ONLY.		#	327
# 73				#	328
# 74	15,1	GROUND COVER CODE TO INDICATE OPEN TRAFFIC-	1-5	#	329

```
#  75          ABLE AREAS NEAR ROAD THAT ALLOWS REST OR ALTER-                    #  330
#  76          NATE PATHS.                                                       #  331
#  77          1=URBAN, 2=CULTIVATED, 3=WOODED, 4=SWAMP,                         #  332
#  78          5=ROAD NET ADJACENT TO MAIN LINK.                                 #  333
#  79                                                                            #  334
#  80     15,2   TERRAIN.  CODE TO INDICATE TERRAIN AND COARSE      1-13         #  335
#  81          INTERVISIBILITY MEASURE.  1=FLAT, 2=HILLY,                        #  336
#  82          3=MOUNTAINOUS, 10=HIGH TERRAIN IMMEDIATELY TO THE                 #  337
#  83          WEST, AND ANY OF THE FIRST THREE PLUS 10.                         #  338
#  84                                                                            #  339
#  85     16,1   BRIDGE.0=NONE, 1=SMALL, 2=MEDIUM, 3=LARGE.        0-3          #  340
#  86                                                                            #  341
#  87     16,2   CITY.  0=NONE, 1=SMALL, 2=MEDIUM, 3=LARGE         0-3          #  342
#  88                                                                            #  343
#  89     17-21  IDENTICAL TO WORDS 12-16 BUT APPLICABLE TO THE                  #  344
#  90          SECOND ADJACENT NODE, IF PRESENT.                                 #  345
#  91                                                                            #  346
#  92     22-26  IDENTICAL TO WORDS 12-16 BUT APPLICABLE TO THE                  #  347
#  93          THIRD ADJACENT NODE, IF PRESNET.                                  #  348
#  94                                                                            #  349
#  95     27-31  IDENTICAL TO WORDS 12-16 BUT APPLICABLE TO THE                  #  350
#  96          FOURTH ADJACENT NODE, IF PRESENT.                                 #  351
#  97                                                                            #  352
#  98     32,1   RISK MEASURE FOR THE LINK BETWEEN THIS NODE AND    0-127        #  353
#  99          ITS PREDECESSOR.  RISK IS TAKEN TO BE THE SUM OF                  #  354
#  00          GROUND COVER, TERRAIN, BRIDGE, AND CITY CODES.                    #  355
#  01                                                                            #  356
#  02     32,2   USED BY THE PATH ALGORITHM TO REMEMBER HOW MANY    0-255        #  357
#  03          MINUTES A UNIT PARKED AT THIS NODE.                               #  358
#                                                                                #
################################################################################
****** TOO MANY LINES IN SEGMENT
```

TERRAIN

```
#                                                                                #
#  1        THIS DATA BASE SUPPORTS THE TERRAIN MODEL FOR PATH CALCULATIONS.     #      360
#  2    THERE ARE ACTUALLY THREE DATA BASES INVOLVED, CROSS-COUNTRY MOVEMENT,    #      361
#  3    SUMMER CONCEALMENT, AND WINTER CONCEALMENT.                              #      362
#  4        EACH DATA BASE SUPPLIES DATA FOR A SPECIFIC TYPE OF PATH CALCULA-    #      363
#  5    TION.  CROSS-COUNTRY YIELDS DATA FOR FASTEST MOVEMENT, AND THE TWO CON-  #      364
#  6    CEALMENT DATA BASES GIVE INFORMATION ABOUT THE RISK OF A PATH.           #      365
#  7        THESE DATA BASES WERE CREATED BY PLACING A 93 X 88 SQUARE GRID OVER  #      366
#  8    THE DATA MAPS SUPPLIED BY THE ARMY.  EACH SQUARE WAS THEN ASSIGNED A     #      367
#  9    VALUE CORRESPONDING TO THE AREA WHICH THE SQUARE COVERED.  THESE VAL-    #      369
# 10    UES WERE TRANSFERRED TO COMPUTER CARDS WHICH WERE THEN USED TO CREATE    #      369
# 11    THREE DATA FILES ON DISK, 'CCM.DAT', 'SCON.DAT', AND 'WCON.DAT'.         #      370
# 12        BECAUSE OF THE REGULARITY OF THE SQUARES, EACH SQUARE MAY BE RE-     #      371
# 13    GARDED AS A NODE WITH EIGHT NEIGHBORS (UNLESS IT LIES ON AN EDGE). THUS, #      372
# 14    EACH MAP IS REPRESENTED BY 8184 NODES.  FOR ANY NODE, NO MORE INFORMA-   #      373
# 15    TIONS IS NEEDED, SINCE ALL LINK ATTRIBUTES ARE KNOWN.                    #      374
# 16        THERE IS, HOWEVER, PATH INFORMATION WHICH MUST BE STORED DURING      #      375
# 17    A CALCULATION, BUT THERE SIMPLY ISN'T ENOUGH MEMORY TO ALLOCATE 5 WORDS  #      376
# 18    FOR EACH OF THE 8184 NODES.  THUS, WHEN USING THE TERRAIN MODEL FOR      #      377
# 19    PATH CALCULATIONS, THERE IS AN ADDITIONAL DATA BASE CALLED THE 'WORKING  #      378
# 20    LIST'.  AS A NEW NODE IS INCLUDED IN THE CALCULATION, A NEW ENTRY TO     #      379
# 21    THE WORKING LIST IS MADE.  SINCE NOT ALL NODES WILL BE INCLUDED, THE     #      380
# 22    WORKING LIST'S SIZE IS CONSIDERABLY SMALLER THAN 8184 * 5 WORDS, AND     #      381
# 23    FITS IN MAIN MEMORY WITH EASE.                                           #      382
# 24        ANOTHER SPACE-SAVING DEVICE IS EMPLOYED.  THE CODE FOR ANY SQUARE    #      383
# 25    IS NEVER GREATER THAN 15, SO DATA FOR FOUR NODES MAY BE PLACED IN ONE    #      384
# 26    WORD OF MEMORY, THUS 8184 CODES USE ONLY 2046 WORDS.                     #      385
#                                                                                #
##################################################################################
```

CROSS-COUNTRY AND CONCEALMENT CODE DESCRIPTIONS

THESE CODES COME DIRECTLY FROM THE MAPS PREPARED BY THE TERRAIN ANALYSIS CENTER, US ARMY ENGINEER TOPOGRAPHIC LABORATORIES, FORT BELVOIR, VIRGINIA, IN SEPTEMBER, 1977.

CROSS-COUNTRY MOVEMENT

CODE	ESTIMATED MAX. SPEED OF XM-1 TANK (MPH)	MANEUVERABILITY DESCRIPTOR
1	>= 25	EXCELLENT
2	20 - 25	VERY GOOD
3	10 - 20	GOOD
4	5 - 10	FAIR
5	< 5	POOR
6	PASSAGE BLOCKED	NO GO
7	BUILT-UP AREA (TRAVEL OFF ROADS NOT EVALUATED)	

CONCEALMENT

CODE	PERCENT CHANCE OF BEING OBSERVED FROM THE AIR
8	0 - 25 (BEST CONCEALMENT)
9	25 - 50
10	50 - 75
11	75 - 100 (POOREST CONCEALMENT)

UNITS

```
#                                                                                #
#  1        THIS IS A TEMPORARY DATA BASE, THAT IS, A DATA BASE WHICH MAY       #   414
#  2   CHANGE FROM USER TO USER, AND, IN FACT, FROM ONE PATH CALCULATION TO     #   415
#  3   ANOTHER, IF DESIRED.  BECAUSE OF ITS TEMPORARY NATURE, IT EXISTS ONLY    #   416
#  4   IN MAIN MEMORY; THERE IS NO UNITS DATA BASE ON DISK.                     #   417
#  5        THE UNIT←VECTOR IS THE SOLE CONSTITUENT WITHIN THIS DATA BASE.  IT  #   418
#  6   CONTAINS INFORMATION PERTAINING TO A UNIT, SUCH AS ITS NAME, STARTING    #   419
#  7   LOCATION, DESTINATION LOCATION, PRESENT LOCATION, PRIORITY, START TIME,  #   420
#  8   AND TYPE CODE.  EACH LOCATION IS ENTERED BY THE USER AS A MAP←NUMBER,    #   421
#  9   NODE←NUMBER PAIR, WHERE THE MAP←NUMBER IS THE NUMBER OF THE 1:50000 MAP  #   422
# 10   ON WHICH THE NODE LIES, AND THE NODE←NUMBER IS THAT NUMBER ARBITRARILY   #   423
# 11   ASSIGNED WHEN THE NODE←VECTOR DATA BASE WAS CONSTRUCTED.  HOWEVER, FOR   #   424
# 12   PURPOSES OF STORAGE, THESE LOCATIONS ARE REPRESENTED INTERNALLY AS IN-   #   425
# 13   DEXES INTO THE NODE←VECTOR ARRAY.  PRIORITY IS A NUMBER REPRESENTING     #   426
# 14   A UNIT'S IMPORTANCE WITH RESPECT TO OTHER UNITS.  START TIME IS A VALUE  #   427
# 15   ENTERED BY A PATH ALGORITHM, NOT THE USER, AND REPRESENTS MINUTES OF     #   428
# 16   TIME.  TYPE CODE IS A NUMBER WHICH STANDS FOR THE PARTICULAR TYPE OF     #   429
# 17   UNIT BEING CONSIDERED, SUCH AS A RIFLE BATTALION, OR A MOTORIZED DIVI-   #   430
# 18   SION.                                                                   #   431
# 19        THIS DATA BASE MAY BE RECREATED EACH TIME A USER WISHES TO PER-    #   432
# 20   FORM A PATH CALCULATION, AND MAY CONTAIN ANYWHERE FROM 1 TO 60 SEPARATE  #   433
# 21   UNITS.  IF NO UNITS ARE ESTABLISHED BY THE USER, THE PATH ALGORITHMS     #   434
# 22   WILL NOT FUNCTION, AND AN ERROR MESSAGE PRINTED.                        #   435
#                                                                                #
##################################################################################
```

UNIT←VECTOR FORMAT

1	WORD #	BYTE 1	BYTE 2	437
2	1	FIRST LETTER	SECOND LETTER	438
3	2	THIRD LETTER	FOURTH LETTER	439
4	3	FIFTH LETTER	SIXTH LETTER	440
5	4	SEVENTH LETTER	EIGHTH LETTER	441
6	5	STARTING LOCATION		442
7	6	DESTINATION LOCATION		443
8	7	PRESENT LOCATION		444
9	8	PRIORITY	TYPE CODE	445
10	9	START TIME		446
11	10	SPARE		447

UNIT←VECTOR DESCRIPTION

	WORD #	DESCRIPTION AND USE	RANGE
1	1-4	THIS IS THE UNIT'S NAME IN CHARACTER FORMAT. THERE MAY BE AT MOST EIGHT CHARACTERS.	ANY ALLOWABLE CHARACTER
2	5	AN INDEX INTO THE NODE←VECTOR ARRAY. THUS, THIS WORD SAYS THAT IF THE INDEX EQUALS I, THE PROPER NODE IS THE I TH NODE IN THE ARRAY.	1-400
3	6	AN INDEX INTO THE NODE←VECTOR ARRAY, INDICATING THE DESTINATION OF THIS UNIT.	1-400
4	7	AN INDEX INTO THE NODE←VECTOR ARRAY, INDICATING WHERE THE UNIT IS AT THIS MOMENT IN THE CALCULATION.	1-400
5	8,1	PRIORITY OF THIS UNIT. 1 IS THE HIGHEST, 127 IS THE LOWEST. 0 INDICATES THAT THE PRIORITY HAS NOT BEEN ESTABLISHED.	0-127
6	8,2	CODE INDICATING GENERAL TYPE OF UNIT.	0-255
7	9	TIME AT WHICH UNIT IS KNOWN TO BE AT EITHER ITS STARTING OR FINISHING NODE. THIS DEPENDS UPON MOVEMENT TYPE.	0-15839
8	10	SPARE	

SOLUTIONS

```
#                                                                                #
#  1        ANOTHER TEMPORARY DATA BASE, THIS IS CREATED BY THE PATH ALGORITHM    #   481
#  2    AS IT RECONSTRUCTS A ROUTE OF TRAVEL.  EACH SOLUTION←VECTOR SO CREATED    #   482
#  3    CONTAINS INFORMATION ABOUT A PARTICULAR NODE; ITS NODE←VECTOR INDEX,       #   483
#  4    TIME IN, TIME OUT, LANE NUMBER, AND PARK TIME, AND FURTHER DIAGNOSTIC     #   484
#  5    DATA; A POINTER TO ANOTHER SOLUTION←VECTOR WHICH CONSIDERS THE SAME       #   485
#  6    NODE←VECTOR (IF NONE, THE POINTER IS NULL), AND WHETHER OR NOT THIS SOL-  #   486
#  7    UTION←VECTOR SHOULD BE USED BY THE PATH DECONFLICTING LOGIC.              #   487
#  8        A NODE←VECTOR INDEX IS A POINTER TO A NODE←VECTOR IN MAIN MEMORY.     #   488
#  9    TIME IN AND TIME OUT INDICATE A TIME PERIOD DURING WHICH THE NODE←VEC-    #   489
# 10    TOR IS BUSY.  LANE NUMBER KEEPS TRACK OF WHICH LANE THIS UNIT USED TO     #   490
# 11    TRAVEL THROUGH THIS NODE.  THIS ENABLES MULTIPLE LANE ROADS TO BE USED    #   491
# 12    BY MORE THAN ONE UNIT AT A TIME.  PARK TIME SHOWS HOW MUCH TIME (IN MIN-  #   492
# 13    UTES) THE UNIT HAD TO WAIT AT THIS NODE'S PREDECESSOR BEFORE TRAVELING    #   493
# 14    TO THIS NODE.                                                             #   494
# 15        THE SOLUTION←VECTOR POINTER ALLOWS THE DECONFLICTING LOGIC TO OPER-   #   495
# 16    ATE.  BY FOLLOWING THE POINTERS, IT IS POSSIBLE TO FIND ALL INTERVALS     #   496
# 17    DURING WHICH A PARTICULAR NODE IS BUSY.  IN THIS WAY, DECONFLICTING, THE  #   497
# 18    PROCESS OF MAKING SURE THAT NO TWO UNITS USE A NODE DURING OVERLAPPING    #   498
# 19    TIME INTERVALS, MAY BE ACCOMPLISHED.  THE USER HAS THE OPTION TO INCLUDE  #   499
# 20    OR EXCLUDE THE DECONFLICTING LOGIC.  DATA WITHIN THE SOLUTION←VECTOR      #   500
# 21    REMEMBERS THE USER'S CHOICE.                                              #   501
# 22     THE ROUTE DESCRIBED BY A PATH IS A SERIES OF SOLUTION←VECTORS            #   502
# 23    HAVING A ONE-TO-ONE CORRESPONDENCE WITH THE NODES OF THE PATH.  IF A      #   503
# 24    PATH'S NODES WERE NUMBERED 1-N, ITS SOLUTION←VECTORS WOULD BE NUM-        #   504
# 25    BERED M+1-M+N, ASSUMING M SOLUTION←VECTORS ALREADY EXISTED AT THE TIME    #   505
# 26    THE PATH WAS CALCULATED.  NODE 1 WOULD CORRESPOND TO SOLUTION←VECTOR      #   506
# 27    M+1, NODE 2 WOULD CORRESPOND TO SOLUTION←VECTOR M+2, ETC.                 #   507
#                                                                                #
##################################################################################
```

SOLUTION-VECTOR FORMAT

	WORD #	CONTENTS	
1			509
2			510
3	1	NODE NUMBER	511
4	2	TIME IN	512
5	3	TIME OUT	513
6	4	MULTIPLE USE NODE POINTER TO ENTRY IN THIS LIST.	514
7	5	+,- LANE NUMBER;PARK TIME	515

SOLUTION←VECTOR DESCRIPTION

	WORD #	DESCRIPTION AND USE	RANGE	
1				517
2				518
3	1	INDEX NUMBER OF THE ASSOCIATED NODE←VECTOR.	1-32767	519
4		SEVERAL ROUTES FOR DIFFERENT UNITS MAY POINT		520
5		TO THE SAME NODE←VECTOR.		521
6				522
7	2	THIS IS THE SIMULATED TIME THAT THE UNIT ENTERS THE	0-14400 MIN.	523
8		NODE AS DETERMINED DURING CONSTRUCTION OF THE SOL-		524
9		UTION←VECTOR. TIME UNITS ARE MINUTES.		525
10				526
11	3	THIS IS THE SIMULATED TIME THAT THE UNIT LEAVES THE	0-14400 MIN.	527
12		NODE. IT IS COMPUTED AS A COMPANION WITH THE ABOVE		528
13		WORD #2. THE UNITS ARE IDENTICAL.		529
14				530
15	4	THIS NODE←ARRAY INDEX POINTS TO ANOTHER ENTRY IN	0-32767	531
16		THIS SOLUTION VECTOR LIST WHICH IS ASSOCIATED WITH		532
17		THE SAME NODE←VECTOR. THIS LINK SHOWS THE MULTIPLE USE		533
18		OF A NODE BY SEVERAL ROUTES OF DIFFERENT UNITS. CHECKING		534
19		THE TIMES IN AND OUT (WORDS 2 AND 3) OF THE MULTIPLE USERS		535
20		OF THE NODE WILL INDICATE WHETHER OR NOT A CONFLICT HAS		536
21		OCCURRED.		537
22				538
23	5,1	THIS IS THE LANE NUMBER WHICH THIS UNIT USED TO TRAVEL	1-4	539
24		THROUGH THIS NODE.		540
25				541
26	5,2	TIME IN MINUTES SPENT PARKING AT THE PREDECESSOR NODE	+,- 0-32767 MI	542
27		BEFORE TRAVELING TO THIS NODE.		543
28		A POSITIVE VALUE INDICATES THE VECTOR SHOULD BE CON-		544
29		SIDERED WHEN DE-CONFLICTING. IF NEGATIVE, IT MUST BE		545
30		SKIPPED.		546

ROUTES

```
#                                                                                #
#  1        A PATH HAS BEEN DEFINED TO BE A SERIES OF NODES THROUGH WHICH        #      548
#  2    A UNIT TRAVELS IN ORDER TO GET FROM ONE SPECIFIC NODE TO ANOTHER.  A      #      549
#  3    ROUTE←VECTOR CONTAINS INFORMATION REGARDING A PARTICULAR ROUTE.  THIS     #      550
#  4    INCLUDES A POINTER TO THE UNIT WHICH IS TRAVELING ALONG THE PATH, A      #      551
#  5    POINTER TO THE FIRST SOLUTION←VECTOR FOR THE ROUTE, TOTAL MARCH TIME,    #      552
#  6    TOTAL RISK MEASURE FOR THE PATH, TOTAL PATH DISTANCE, NUMBER OF SOLU-    #      553
#  7    TION←VECTORS IN THE ROUTE, TYPE OF MOVEMENT (FORWARD OR BACKWARD IN      #      554
#  8    TIME), AND INITIAL TIME.                                                 #      555
#  9        THE UNIT POINTER IS AN INDEX INTO THE UNITS DATA BASE, THE SOL-      #      556
# 10    UTION←VECTOR POINTER IS AN INDEX INTO THE SOLUTIONS DATA BASE, AND INIT- #      557
# 11    IAL TIME IS EITHER THE START TIME OR FINISH TIME, AS DICTATED BY THE     #      558
# 12    MOVEMENT TYPE.                                                           #      559
# 13        A ROUTE←VECTOR IS CREATED IMMEDIATELY AFTER THE SOLUTION←VECTORS     #      560
# 14    ARE GENERATED FOR A PARTICULAR ROUTE.  THE ROUTES DATA BASE IS THEN      #      561
# 15    TEMPORARY, AND EXISTS ONLY IN MAIN MEMORY.  THE USER HAS NO CONTROL      #      562
# 16    OVER THE DATA PLACED WITHIN A ROUTE←VECTOR, AND CAN ONLY KNOW ITS        #      563
# 17    CONTENTS THROUGH INDIRECT MEANS.  ONE ROUTINE LISTS UNIT NAMES AND       #      564
# 18    ASSOCIATED ROUTES.  ANOTHER LISTS PATH DIAGNOSTICS, SUCH AS TOTAL        #      565
# 19    TRAVEL TIME, TOTAL DISTANCE, AND TOTAL RISK MEASURE, ALONG WITH OTHER    #      566
# 20    INFORMATION.  EACH OF THESE USES THE ROUTES DATA BASE AS A SOURCE.       #      567
#                                                                                #
```

ROUTE<VECTOR FORMAT

WORD #	CONTENTS	
1		569
2		570
3	UNIT NUMBER	571
4	LIST HEAD POINTER	572
5	TOTAL ROUTE TIME	573
6	TOTAL ROUTE RISK MEASURE	574
7	TOTAL ROUTE DISTANCE	575
8	NUMBER OF SOLUTION<VECTORS IN ROUTE	576
9	MOVEMENT TYPE	577
10	INITIAL TIME	578

ROUTE←VECTOR FORMAT DESCRIPTION

	WORD #	DESCRIPTION AND USE	RANGE	
1				580
2				581
3	1	INDEX TO UNIT DATA BASE, SO THAT WE KNOW	1-60	582
4		WHICH UNIT USES THIS ROUTE.		583
5				584
6	2	INDEX TO THE SOLUTION VECTOR LIST, POINT-	1-5000	585
7		ING TO THE FIRST SOLUTION VECTOR WHICH		586
8		HAS THIS ROUTE NUMBER.		587
9				588
10	3	TOTAL ELAPSED TIME OF MARCH FROM START-	0-15839 MIN.	589
11		ING NODE TO DESTINATION NODE, INCLUDING		590
12		RESTS AND STOPS, IN MINUTES.		591
13				592
14	4	TOTAL CUMMULATIVE RISK MEASURE OF ROUTE	0-32767	593
15				594
16	5	TOTAL ROUTE DISTANCE, IN TENTHS OF KMS.	0-3276.7	595
17				[illegible]
18	6	TOTAL NUMBER OF NODES VISITED	1-400	597
19				598
20	7	CODE INDICATING THAT THIS WAS DETERMINED	0 OR 1	599
21		USING A STARTING TIME (=0) OR AN ARRIVAL		600
22		TIME (=1).		601
23				602
24	8	TIME, IN MINUTES, WHICH THE USER DESIGNATED	0-15839	603
25		AS EITHER A STARTING TIME OR AN ARRIVAL TIME.		604
26		DICTATED BY MOVEMENT TYPE.		605

MAP DIRECTORY

```
#                                                                                #
#  1        THIS DATA BASE KEEPS TRACK OF THE MAPS DATA BASE.  THERE IS ONE      #     607
#  2    ENTRY FOR EACH MAP IN THE DISK FILE.  THIS ENTRY CONTAINS TWO PIECES OF  #     608
#  3    INFORMATION; 1) MAP NUMBER, AND 2) NUMBER OF NODES IN THE MAP.  THE      #     609
#  4    FIRST ENTRY IN THE MAP DIRECTORY CONTAINS A VALUE EQUAL TO THE NUMBER OF #     610
#  5    MAPS WHICH HAVE BEEN ENTERED.  WHEN A MAP'S NODE INFORMATION IS PLACED   #     611
#  6    IN THE DATA BASE, ITS DIRECTORY ENTRY IS PLACED IN THE NEXT AVAILABLE    #     612
#  7    LOCATION OF THE MAP DIRECTORY.  THUS THE SECOND MAP DIRECTORY ENTRY COR- #     613
#  8    RESPONDS TO THE FIRST MAP PLACED IN THE MAPS DATA BASE, THE THIRD ENTRY  #     614
#  9    CORRESPONDS TO THE SECOND MAP, ETC.                                      #     615
#                                                                                #
```

CROSS-COUNTRY DIRECTORY

```
#                                                                                #
#  1        THIS DIRECTORY IS USED TO FIND SPECIFIC MAPS IN THE CROSS-COUNTRY     #      617
#  2    AND CONCEALMENT DISK FILES.  EACH MAP HAS ONE DIRECTORY ENTRY, THIS EN-   #      618
#  3    TRY CONTAINING THE MAP NUMBER, AND THE UTM LATITUDE AND LONGITUDE EX-     #      619
#  4    TREMES OF THE MAP.  THESE ARE THE COORDINATES OF THE LOWER LEFT AND UP-   #      620
#  5    PER RIGHT CORNERS.  THE FIRST ENTRY OF THE DIRECTORY CONTAINS ONLY A      #      621
#  6    COUNTER WHICH KEEPS TRACK OF HOW MANY ENTRIES ARE PRESENT IN THE DIREC-   #      622
#  7    TORY.                                                                     #      623
#                                                                                #
```

PREPARATION FOR USE

PREPARATION

```
REF
PAGE ************************************************************************************
     *                                                                                  *
     *  1    do ONCE BEFORE MAIN-PROGRAM EXECUTION                                     *    626
  27 *  2        CREATE NODE-VECTOR AND MAP-DIRECTORY FILES ON DISK                    *    627
  33 *  3        CREATE CROSS-COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK        *    628
     *  4    enddo ONCE BEFORE MAIN-PROGRAM EXECUTION                                  *    629
     *  5    do AS REQUIRED                                                            *    630
     *  6        AMEND FILES                                                           *    631
     *  7    enddo AS REQUIRED                                                         *    632
     *                                                                                  *
     ************************************************************************************
```

CREATE NODE←VECTOR AND MAP←DIRECTORY FILES ON DISK

```
REF
PAGE ***************************************************************************************
     *                                                                                     *
     *  1    .BEFORE THE FILES MAY BE CREATED, IT IS NECESSARY TO CREATE                   *      634
     *  2    .NODE AND LINK CARDS. INFORMATION FOR THESE CARDS IS VISUALLY                 *      635
     *  3    .DETERMINED FROM THE PARTICULAR L-SERIES MAP BEING CONSIDERED.               *      636
     *  4    .                                                                             *      637
     *  5    do FOR EACH MAP                                                               *      638
     *  6        do FOR EACH NODE                                                          *      639
 28  *  7            CREATE NODE CARD                                                      *      640
     *  8        enddo FOR EACH NODE                                                       *      641
     *  9        do FOR EACH LINK                                                          *      642
 30  * 10            CREATE LINK CARD                                                      *      643
     * 11        enddo FOR EACH LINK                                                       *      644
     * 12    enddo FOR EACH MAP                                                            *      645
     * 13    .                                                                             *      646
     * 14    .NOW THAT THE CARDS ARE READY FOR PROCESSING, CREATE THE NODE←VECTOR          *      647
     * 15    .FILE ON DISK, THAT IS, ESTABLISH NODE←VECTORS USING INFORMATION GAINED       *      648
     * 16    .FROM NODE AND LINK CARDS.                                                    *      649
     * 17    .FIRST, INITIALIZE THE FILES                                                  *      650
     * 18    .                                                                             *      651
     * 19    do ONCE                                                                       *      652
     * 20        CREATE DISK FILE FOR NODE←VECTORS, 'MAPS.DAT'                             *      653
     * 21        CREATE DISK FILE FOR MAP←DIRECTORY, 'MAPDIR.DAT'                          *      654
     * 22    enddo ONCE                                                                    *      655
     * 23    .                                                                             *      656
     * 24    .NOW BUILD THE VECTORS IN MEMORY                                              *      657
     * 25    .                                                                             *      658
     * 26    do UNTIL ALL MAPS ARE PROCESSED                                               *      659
     * 27        do FOR EACH MAP                                                           *      660
     * 28            do UNTIL NO MORE NODE CARDS                                           *      661
     * 29                READ A NODE CARD                                                  *      662
     * 30                PLACE INFORMATION INTO MAIN MEMORY                                *      663
     * 31            enddo UNTIL NO MORE NODE CARDS                                        *      664
     * 32            do UNTIL NO MORE LINK CARDS                                           *      665
```

```
    * 33              READ A LINK CARD... NODE A TO NODE B                          *      666
    * 34              FILL IN LINK INFORMATION TO NODE<VECTOR FOR A                 *      667
    * 35              if NODE<VECTOR FOR B IS ON THIS MAP                           *      668
    * 36                  FILL IN LINK INFORMATION TO NODE<VECTOR FOR B              *      669
    * 37              else CYCLE                                                    *      670
    * 38              endif NODE<VECTOR FOR B IS ON THIS MAP                        *      671
    * 39          enddo UNTIL NO MORE LINK CARDS                                    *      672
    * 40          READ 'MAPDIR.DAT' INTO MEMORY                                     *      673
    * 41          READ NUMBER OF MAPS PROCESSED                                     *      674
    * 42          INCREMENT NUMBER OF MAPS PROCESSED                                *      675
 32 * 43          CREATE MAP<DIRECTORY ENTRY FOR THIS MAP                           *      676
    * 44          WRITE NODE<VECTOR ARRAY FOR THIS MAP TO 'MAPS.DAT'                *      677
    * 45          WRITE MAP<DIRECTORY TO 'MAPDIR.DAT'                               *      678
    * 46      enddo FOR EACH MAP                                                    *      679
    * 47  enddo UNTIL ALL MAPS ARE PROCESSED                                        *      680
    *                                                                               *
    *********************************************************************************
```

CREATE NODE CARD

REF
PAGE

```
*                                                                                *
*  1    .NODE CARDS WILL BE KEY PUNCHED MANUALLY, USING THE FOLLOWING FORMAT FOR  *    682
*  2    .THEIR CONSTRUCTION.                                                      *    683
*                                                                                *
```

NODE CARD FORMAT

	ENTRY	EXAMPLE	FIELD	
1				685
2				686
3	NODE MAP NUMBER. THE	4924	1-5(5)	687
4	1:50000 MAPS WILL BE USED.			688
5	ENTER THE NUMERICAL MAP			689
6	NUMBER, NOT THE LETTERS.			690
7				691
8	NODE NUMBER WITHIN THE MAP.	18	6-9(4)	692
9	A MASTER OVERLAY (PLASTIC)			693
10	FOR EACH MAP MUST BE KEPT			694
11	TO KEEP TRACK OF THE NUMBERS			695
12	AND NODES.			696
13				697
14	NODE LATITUDE IN KM'S FROM	3245	10-16(7)	698
15	THE EQUATOR, READ DIRECTLY			699
16	FROM THE 1:50000 MAP.			700
17				701
18	NODE LONGITUDE IN KM'S FROM	125	17-23(7)	702
19	GREENWICH, READ DIRECTLY			703
20	FROM THE 1:50000 MAP.			704
21				705
22	FIRST ADJACENT NODE MAP #.	4923	24-28(5)	706
23				707
24	FIRST ADJACENT NODE NUMBER	201	29-32(4)	708
25	FOR THE MAP OF THIS ADJACENT			709
26	NODE. ENTER THE NUMBER ONLY.			710
27				711
28	SECOND ADJACENT NODE DATA,		33-41(9)	712
29	ENTER THE NUMBER ONLY.			713
30				714
31	THIRD ADJACENT NODE DATA,		42-50(9)	715
32	ENTER THE NUMBER ONLY.			716
33				717
34	FOURTH ADJACENT NODE DATA,		51-59(9)	718
35	ENTER THE NUMBER ONLY.			719
36				720

# 37	NUMBER OF ADJACENT NODES.	4	60-61(2)	#	721
# 38				#	722
#				#	

CREATE LINK CARD

REF
PAGE

```
*                                                                                *
*  1    . LINK CARDS WILL BE KEY-PUNCHED MANUALLY, USING THE                     *   724
*  2    .FOLLOWING FORMAT.                                                       *   725
*                                                                                *
**********************************************************************************
```

LINK CARD FORMAT

	ENTRY	EXAMPLE	FIELD	
1				727
2				728
3	FIRST NODE MAP NUMBER.	4924	1-5(5)	729
4				730
5	FIRST NODE NUMBER WITHIN MAP.	13	6-9(4)	731
6				732
7	SECOND NODE MAP NUMBER.	4924	10-14(5)	733
8				734
9	SECOND NODE NUMBER WITHIN	17	15-18(4)	735
10	MAP.			736
11				737
12	LINK DISTANCE IN TENTHS OF A	18.7	19-23(5)	738
13	KILOMETER BETWEEN FIRST AND			739
14	SECOND NODES.			740
15				741
16	NUMBER OF EQUIVALENT LANES.	3	24-26(3)	742
17				743
18	ROAD TYPE CODE.	3	27-28(2)	744
19				745
20	GROUND COVER CODE.	3	29-30(2)	746
21				747
22	TERRAIN CODE.	2	31-32(2)	748
23				749
24	BRIDGE CODE.	0	33-34(2)	750
25				751
26	CITY CODE.	1	35-36(2)	752

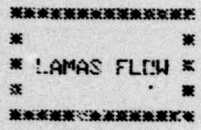
```
     CREATE MAP←DIRECTORY ENTRY FOR THIS MAP


REF
PAGE ***************************************************************************************************
     *                                                                                                 *
     *  1    .AT THIS POINT IN THE PROGRAM, THIS MAP'S NODE←VECTORS HAVE                             *   754
     *  2    .BEEN READ INTO CORE, AND ALL PERTINENT INFORMATION HAS BEEN                            *   755
     *  3    .PLACED IN THE PROPER AREA. THE MAP←NUMBER IS IN THE FIRST                              *   756
     *  4    .WORD OF THE NODE←VECTOR ARRAY, AND THE NUMBER OF NODES IN                              *   757
     *  5    .THIS MAP IS KNOWN FROM A COUNTER WHICH WAS INCREMENTED EVERY TIME                      *   758
     *  6    .A NEW NODE CARD WAS READ.                                                              *   759
     *  7    .                                                                                       *   760
     *  8    MAP←DIRECTORY ENTRY(1,NUMBER OF MAPS)=MAP←NUMBER                                        *   761
     *  9    MAP←DIRECTORY ENTRY(2,NUMBER←OF←MAPS)=NUMBER OF NODES                                   *   762
     *                                                                                                 *
     ***************************************************************************************************
```

CREATE CROSS-COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK

```
REF
PAGE ***************************************************************************
     *                                                                           *
     *  1    .BEFORE THESE THREE FILES MAY BE ESTABLISHED, THE DATA MUST BE      *  764
     *  2    .PLACED ON COMPUTER CARDS. THIS DATA IS OBTAINED BY PLACING A       *  765
     *  3    .93 X 88 GRID OVER THE APPROPRIATE MAP. EACH SQUARE OF THE GRID     *  766
     *  4    .IS ASSIGNED A VALUE EQUAL TO THE CODE WHICH MOST DESCRIBES THE     *  767
     *  5    .AREA. THAT IS, IF THE AREA UNDERNEATH THE SQUARE IS PARTIALLY      *  768
     *  6    .CODE 4, AND PARTIALLY CODE 5, THE VALUE ASSIGNED IS EITHER 4 OR    *  769
     *  7    .5, DEPENDING UPON WHICH APPEARS TO OCCUPY THE MAJORITY OF THE      *  770
     *  8    .SQUARE. IF MORE THAN 2 CODES ARE COVERED, THEN THE CODE CHOSEN     *  771
     *  9    .IS THE ONE WHICH DOMINATES (SEEMS TO BE GREATER THAN THE OTHERS    *  772
     * 10    .TAKEN INDIVIDUALLY) THE SQUARE.                                    *  773
     * 11    .                                                                   *  774
     * 12    do FOR EACH MAP                                                     *  775
     * 13        CREATE DATA CARDS                                               *  776
     * 14    enddo FOR EACH MAP                                                  *  777
     * 15    .                                                                   *  778
     * 16    .WITH THE CARDS ALL MADE, IT IS NOW A MATTER OF READING THE CARDS   *  779
     * 17    .INTO MEMORY, CONVERTING THE VALUES AS APPROPRIATE, MAKING A DIR-   *  780
     * 18    .ECTORY ENTRY, AND WRITING THE RESULTS OUT TO DISK.                 *  781
     * 19    .                                                                   *  782
     * 20    do ONCE                                                             *  783
     * 21        CREATE DISK FILE FOR CROSS-COUNTRY MAPS, 'CCM.DAT'              *  784
     * 22        CREATE DISK FILE FOR WINTER CONCEALMENT MAPS, 'WCON.DAT'        *  785
     * 23        CREATE DISK FILE FOR SUMMER CONCEALMENT MAPS, 'SCON.DAT'        *  786
     * 24        CREATE DISK FILE FOR DIRECTORY, 'CCMDIR.DAT'                    *  787
     * 25    enddo ONCE                                                          *  788
     * 26    .                                                                   *  789
     * 27    .BECAUSE THE ALLOWABLE CODES HAVE THE VALUES 1-7 FOR CROSS-COUN-    *  790
     * 28    .TRY AND 8-11 FOR CONCEALMENT, ONLY FOUR BITS ARE NEEDED TO STORE   *  791
     * 29    .THIS INFORMATION. A GOOD SPACE SAVING MAY BE MADE IF THE VALUES    *  792
     * 30    .ARE PACKED FOUR TO A WORD. DOING THIS, THOUGH, MEANS THAT, IN      *  793
     * 31    .ORDER TO DEAL WITH COMPLETE ROWS OF DATA, FOUR ROW OF CARDS        *  794
     * 32    .SHOULD BE PROCESSED AT ONE TIME.                                   *  795
```

```
* 33    .                                                                    *     796
* 34    do FOR EACH MAP                                                      *     797
* 35        do UNTIL ALL ROWS ARE READ IN                                    *     798
* 36            .                                                            *     799
* 37            .SINCE EACH ROW HAS 93 ELEMENTS, AND A COMPUTER CARD HAS ONLY *     800
* 38            .72 COLUMNS, EACH ROW USES TWO CARDS.                        *     801
* 39            .                                                            *     802
* 40            READ FOUR ROWS WORTH OF CARDS (EIGHT ALTOGETHER)             *     803
* 41            PACK THESE ROWS FOUR VALUES TO A WORD                        *     804
* 42        enddo UNTIL ALL ROWS ARE READ IN                                 *     805
* 43        .                                                                *     806
* 44        .ALL DATA IS NOW IN MEMORY. WRITE IT OUT TO PROPER DATA FILE.    *     807
* 45        .                                                                *     808
* 46        WRITE DATA TO DISK                                               *     809
* 47        .                                                                *     810
* 48        .MAKE DIRECTORY ENTRY                                            *     811
* 49        .                                                                *     812
* 50        PROMPT FOR MAP#NUMBER                                            *     813
* 51        READ INPUT                                                       *     814
* 52        PLACE INPUT AT NEXT AVAILABLE PLACE IN COM.DIR ...ONLY DO THIS IF CROSS-COUNTRY MAP  /*  815
*               BEING PROCESSED.                                             *
* 53        PROMPT FOR UTM LATITUDE AND LONGITUDE EXTREMES                   *     817
* 54        PLACE INPUT AT NEXT AVAILABLE PLACE IN COM.DIR ...SAME ADMONITION AS BEFORE.  *  818
* 55        .                                                                *     820
* 56        .WRITE DIRECTORY TO DISK                                         *     821
* 57        .                                                                *     822
* 58        WRITE DIRECTORY TO DISK                                          *     823
* 59    enddo FOR EACH MAP                                                   *     824
*                                                                            *
******************************************************************************
```


LAMAS FLOW

LAMAS

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    do UPON MAIN<PROGRAM EXECUTION                                         *  827
     *  2        do FOREVER                                                         *  828
     *  3            DISPLAY MAIN FUNCTION MENU                                     *  829
     *  4            WAIT FOR USER INPUT                                            *  830
     *  5            do CASE OF :                                                   *  831
     *  6 ROAD<NETWORK:                                                             *  832
  36 *  7                IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE NODE<VECTOR DATA BASE  *  833
     *  8 TERRAIN<MODEL:                                                           *  834
  66 *  9                IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE CROSS-COUNTRY DATA BASE  *  835
     * 10                CONCEALMENT DATA BASES                                     *  836
     * 11 EXIT:                                                                    *  837
     * 12                EXIT THIS SYSTEM                                           *  838
     * 13            enddo CASE OF                                                 *  839
     * 14        enddo FOREVER                                                     *  840
     * 15    enddo UPON MAIN<PROGRAM EXECUTION                                     *  841
     *                                                                              *
     ********************************************************************************
```

IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE NODE-VECTOR DATA BASE

```
REF
PAGE ***************************************************************************************
     *                                                                                     *
     *  1    do UPON FUNCTION REQUEST                                                      *      843
     *  2        do FOREVER                                                                *      844
     *  3            DISPLAY FUNCTION MENU                                                 *      845
     *  4            WAIT FOR USER INPUT                                                   *      846
     *  5            do CASE OF                                                            *      847
     *  6 INITIALIZE:                                                                      *      848
  37 *  7                PREPARE FOR ROAD NETWORK PATH CALCULATIONS                        *      849
     *  8 CALCULATION:                                                                     *      850
  45 *  9                PATH DETERMINATION AND DISPLAY                                    *      851
     * 10 EXIT:                                                                           *      852
     * 11                return TO CALLING PROGRAM                                         *      853
     * 12            enddo CASE OF                                                        *      854
     * 13        enddo FOREVER                                                            *      855
     * 14    enddo UPON FUNCTION REQUEST                                                  *      856
     *                                                                                     *
     ***************************************************************************************
```

PREPARE FOR ROAD NETWORK PATH CALCULATIONS

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    do UPON FUNCTION REQUEST                                               *  858
     *  2        do FOREVER                                                         *  859
     *  3            DISPLAY FUNCTION MENU                                          *  860
     *  4            WAIT FOR USER INPUT                                            *  861
     *  5            do CASE OF                                                     *  862
     *  6 MAPS:                                                                     *  863
 38  *  7                INITIALIZE MAP←NUMBERS AND SCREEN EXTREMES                 *  864
     *  8 UNITS:                                                                    *  865
 42  *  9                INITIALIZE UNIT←VECTORS                                    *  866
     * 10 FEBA:                                                                    *  867
 43  * 11                ESTABLISH A FORWARD EDGE OF BATTLE AREA                    *  868
     * 12 EXIT:                                                                    *  869
     * 13                return TO CALLING PROGRAM                                  *  870
     * 14            enddo CASE OF                                                  *  871
     * 15        enddo FOREVER                                                      *  872
     * 16    enddo UPON FUNCTION REQUEST                                            *  873
     *                                                                              *
     ********************************************************************************
```

INITIALIZE MAP←NUMBERS AND SCREEN EXTREMES

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    do UPON FUNCTION REQUEST                                               *      875
  39 *  2       ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS               *      876
     *  3       do FOR EACH NODE IN MEMORY                                          *      877
     *  4          do FOR EACH ADJACENT NODE                                        *      878
     *  5             if ADJACENT NODE'S MAP←NUMBER IS IN THE DIRECTORY             *      879
  41 *  6                CHANGE ADJACENT NODE'S MAP←NUMBER TO INDEX                 *      880
     *  7             endif ADJACENT NODE'S MAP←NUMBER                              *      881
     *  8          enddo FOR EACH ADJACENT NODE                                     *      882
     *  9       enddo FOR EACH NODE IN MEMORY                                       *      883
     * 10       PROMPT FOR ENTRY OF SCREEN EXTREMES OF LATITUDE AND LONGITUDE       *      884
     * 11       READ INPUT                                                          *      885
     * 12       PLACE INPUT IN COMMON STORAGE                                       *      886
     * 13       return TO CALLING PROGRAM                                           *      887
     * 14    enddo UPON FUNCTION REQUEST                                            *      888
     *                                                                              *
     ********************************************************************************
```

ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . THIS DIRECTORY CONSISTS OF A NUMBER OF ENTRIES, ONE FOR EACH         *      890
     *  2    .MAP WHICH THE USER WISHES TO EMPLOY. THE FIRST DIRECTORY ENTRY CORRE-  *      891
     *  3    .SPONDS TO THE FIRST MAP←NUMBER REQUESTED BY THE USER, AND SO ON FOR    *      892
     *  4    .EACH ENTRY. THERE ARE THREE ELEMENTS IN EACH ENTRY, 1) THE MAP←NUMBER, *      893
     *  5    .2) THE NUMBER OF NODES IN THE MAP, AND 3) THE NUMBER OF NODES ALREADY  *      894
     *  6    .PRESENT IN MEMORY.                                                     *      895
     *  7    . GIVEN THIS MAP←DIRECTORY, IT IS A SIMPLE MATTER TO READ A MAP'S       *      896
     *  8    .NODES INTO MAIN MEMORY, SINCE WE KNOW HOW MANY TO READ IN,             *      897
     *  9    .MAP←DIRECTORY WORD 2, AND WHERE TO START PUTTING THEM, MAP←DIRECTORY   *      898
     * 10    .WORD 3 +1.                                                             *      899
     * 11    .                                                                       *      900
     * 12    READ DISK FILE MAP←DIRECTORY INTO MAIN MEMORY                           *      901
     * 13    PROMPT FOR ENTRY OF MAP←NUMBERS                                         *      902
     * 14    READ USER INPUT OF MAP←NUMBERS TO BE PROCESSED                          *      903
     * 15    do FOR EACH MAP←NUMBER ENTERED BY USER                                  *      904
     * 16        if MAP←NUMBER IS IN THE DISK FILE MAP←DIRECTORY...AS THE I TH ENTRY *      905
     * 17            SET FIRST WORD OF MAIN MEMORY MAP←DIRECTORY TO I                *      906
     * 18            SET SECOND WORD TO NUMBER OF NODES IN THIS MAP                  *      907
     * 19            SET THIRD WORD TO NUMBER OF NODES ALREADY PRESENT IN MEMORY     *      908
  40 * 20            READ THIS MAP'S NODES INTO MAIN MEMORY                          *      909
     * 21            RESET FIRST WORD TO THIS MAP←NUMBER                             *      910
     * 22        else MAP←NUMBER IS NOT IN THE DATA BASE                             *      911
     * 23            PRINT ERROR MESSAGE                                             *      912
     * 24            LET THE USER RE-ENTER THE MAP←NUMBER                            *      913
     * 25            cycle                                                           *      914
     * 26        endif MAP←NUMBER IS IN THE DISK FILE MAP←DIRECTORY                  *      915
     * 27    enddo FOR EACH MAP←NUMBER ENTERED BY USER                               *      916
     *                                                                              *
     ********************************************************************************
```

READ THIS MAP'S NODES INTO MAIN MEMORY

```
REF
PAGE ************************************************************************
     *                                                                      *
     *  1    . AT THIS POINT IN THE PROGRAM, THE MAP←DIRECTORY IN MAIN MEMORY *      918
     *  2    .HAS ITS AS ELEMENTS 1) THE INDEX I, INDICATING THAT THIS MAP IS THE *      919
     *  3    .I TH ONE IN THE DISK FILE MAP←DIRECTORY, 2) NUMBER OF NODES IN THIS MAP *      920
     *  4    .AND 3) NUMBER OF NODES ALREADY PRESENT IN MEMORY. IN ORDER TO *      921
     *  5    .READ THIS MAP'S NODES INTO MEMORY SO THAT NO SPACE IS WASTED, *      922
     *  6    .WE NEED ONLY CONSIDER THE INFORMATION JUST LISTED. TO READ DATA *      923
     *  7    .FROM DISK TO MEMORY WE NEED TO KNOW HOW MANY BYTES WILL BE MOVED. *      924
     *  8    .THIS IS JUST WORD 2 MULTIPLIED BY 64 (32 WORDS * 2 BYTES PER WORD). WE *      925
     *  9    .ALSO MUST KNOW EXACTLY WHERE THE DATA WILL BE PLACED IN MEMORY, THAT *      926
     *  10   .IS, THE NEXT AVAILABLE SPACE IN THE NODE←VECTOR ARRAY. *      927
     *  11   .THIS IS SIMPLY WORD 3 PLUS ONE. FINALLY, WE MUST KNOW FROM WHAT *      928
     *  12   .DISK BLOCK IS THE DATA COMING. THIS IS I-1 MULTIPLIED BY 10, SINCE *      929
     *  13   .THE FIRST MAP STARTS AT BLOCK ONE, THE SECOND AT BLOCK 11, ETC. *      930
     *  14   .NOTE THAT NO MAP HAS MORE THAN 80 NODE←VECTORS, EACH VECTOR *      931
     *  15   .HAVING 32 WORDS, SO THERE ARE 8 NODE←VECTORS TO A DISK BLOCK OF *      932
     *  16   .256 WORDS. *      933
     *  17   . *      934
     *  18   BYTES = (MAP←DIRECTORY WORD 2) * 64 *      935
     *  19   INDEX = (MAP←DIRECTORY WORD 3) + 1 *      936
     *  20   STARTING BLOCK = ((MAP←DIRECTORY WORD 1) -1) * 10 *      937
     *  21   PERFORM DISK READ (BYTES, INDEX, STARTING BLOCK) *      938
     *                                                                      *
     ************************************************************************
```

CHANGE ADJACENT NODE'S MAP←NUMBER TO INDEX

```
REF
PAGE ***********************************************************************
     *                                                                     *
     *  1    .IN ORDER TO ACCESS A NODE QUICKLY, IT IS NECESSARY THAT ONE NOT  *  940
     *  2    .HAVE TO USE THE MAP←DIRECTORY EACH TIME A REFERENCE IS MADE.     *  941
     *  3    .THUS, CONSIDER THE NODE←VECTOR ARRAY AS A LIST OF NODES,         *  942
     *  4    .INDEXABLE BY INTEGERS, SUCH THAT THE FIRST NODE←VECTOR HAS INDEX=1,  *  943
     *  5    .THE SECOND HAS INDEX=2, AND SO ON FOR THE ENTIRE LIST. THEN, IF  *  944
     *  6    .WE KNOW THE MAP←NUMBER AND THE NODE←NUMBER OF A PARTICULAR NODE←VECTOR,  *  945
     *  7    .THEN THIS INDEX MAY BE CALCULATED ONCE, AND SAVED. THE MOST LOGICAL  *  946
     *  8    .PLACE TO SAVE IT IS IN IT'S ADJACENT MAP←NUMBER LOCATION.        *  947
     *  9    . THIS CALCULATION IS SIMPLE, AND DEPENDS ON OUR KNOWLEDGE OF JUST  *  948
     *  10   .TWO THINGS, 1) FOR THE ADJACENT NODE'S MAP←NUMBER, THE NUMBER OF NODES  *  949
     *  11   .ALREADY IN CORE, AND 2) FOR THE ADJACENT NODE, ITS NODE←NUMBER. BOTH  *  950
     *  12   .OF THESE PIECES OF INFORMATION ARE READILY AVAILABLE, THE FIRST IS JUST  *  951
     *  13   .THE THIRD WORD OF THE ADJACENT NODE'S MAP←DIRECTORY ENTRY, AND    *  952
     *  14   .THE SECOND IS IN THE NODE←VECTOR.                                *  953
     *  15   .                                                                 *  954
     *  16   ADJACENT MAP←NUMBER = (ADJACENT NODE'S MAP←DIRECTORY WORD 3) + ADJACENT NODE←NUMBER  *  955
     *                                                                     *
     ***********************************************************************
```

INITIALIZE UNIT←VECTORS

```
REF
PAGE ***************************************************************************************************
     *                                                                                                 *
     *  1    . THIS DATA BASE IS COMPLETELY USER DEFINED AT EXECUTION TIME.                           *      958
     *  2    .THE USER SPECIFIES UNIT NAME, UNIT STARTING AND DESTINATION LO-                         *      959
     *  3    .CATIONS, UNIT PRIORITY, AND UNIT TYPE CODE.                                             *      960
     *  4    .                                                                                        *      961
     *  5    do UPON FUNCTION REQUEST                                                                 *      962
     *  6        ASK IF USER WISHES TO REINITIALIZE OR ADD ON TO EXISTING STRUCTURE                    *      963
     *  7        if USER WANTS TO REINITIALIZE                                                        *      964
     *  8            ERASE ALL UNIT←VECTORS                                                           *      965
     *  9            SET UNIT←COUNTER TO ZERO                                                         *      966
     * 10        endif USER WANTS TO REINITIALIZE                                                     *      967
     * 11        .                                                                                    *      968
     * 12        .PERFORM THE FOLLOWING AS MANY AS SIXTY TIMES                                        *      969
     * 13        .                                                                                    *      970
     * 14        do UNTIL UNIT←COUNTER EQUALS SIXTY                                                   *      971
     * 15            .                                                                                *      972
     * 16            .PROMPT FOR THE FOLLOWING: UNIT NAME, STARTING LOCATION, DESTINATION,            *      973
     * 17            .PRIORITY, AND TYPE CODE. ERROR CHECKING IS BE PERFORMED ON ALL INPUT            *      974
     * 18            .SAVE THE UNIT NAME. INPUT FOR THE START AND FINISH LOCATIONS IS IN THE          *      975
     * 19            .FORM MAP←NUMBER, NODE←NUMBER. ANY TIME THERE IS AN ERROR, THE PROMPT            *      976
     * 20            .WILL BE REPEATED UNTIL VALID INPUT IS RECEIVED.                                 *      977
     * 21            .                                                                                *      978
     * 22            PROMPT USER TO ENTER UNIT NAME                                                   *      979
     * 23            if UNIT NAME IS NULL                                                             *      980
     * 24                return TO CALLING PROGRAM                                                    *      981
     * 25            else INPUT IS VALID CHARACTERS                                                   *      982
     * 26                INCREMENT UNIT←COUNTER                                                       *      983
     * 27                ASSIGN INPUT TO FIRST FOUR WORDS OF NEXT AVAILABLE UNIT←VECTOR               *      984
     * 28                PROMPT FOR STARTING LOCATION                                                 *      985
     * 29                if STARTING LOCATION IS INVALID                                              *      986
     * 30                    PRINT ERROR MESSAGE                                                      *      987
     * 31                    TRY AGAIN                                                                *      988
     * 32                else INPUT IS VALID                                                          *      989
```

```
* 33                  ASSIGN INPUT TO UNIT←VECTOR WORD 5 (STARTING LOCATION)                    *   990
* 34                  ASSIGN INPUT TO UNIT←VECTOR WORD 7 (PRESENT LOCATION)                     *   991
* 35                  PROMPT FOR DESTINATION                                                    *   992
* 36                  if DESTINATION IS INVALID                                                 *   993
* 37                      PRINT ERROR MESSAGE                                                   *   994
* 38                      TRY AGAIN                                                             *   995
* 39                  else INPUT IS VALID                                                       *   996
* 40                      ASSIGN INPUT TO UNIT←VECTOR WORD 6 (DESTINATION)                      *   997
* 41                      PROMPT FOR PRIORITY                                                   *   998
* 42                      if PRIORITY IS INVALID                                                *   999
* 43                          PRINT ERROR MESSAGE                                               *  1000
* 44                          TRY AGAIN                                                         *  1001
* 45                      else INPUT IS VALID                                                   *  1002
* 46                          ASSIGN INPUT TO FIRST BYTE OF UNIT←VECTOR WORD 8                  *  1003
* 47                          PROMPT FOR TYPE CODE                                              *  1004
* 48                          if TYPE CODE IS INVALID                                           *  1005
* 49                              PRINT ERROR MESSAGE                                           *  1006
* 50                              TRY AGAIN                                                     *  1007
* 51                          else INPUT IS VALID                                               *  1008
* 52                              ASSIGN INPUT TO SECOND BYTE OF UNIT←VECTOR WORD 8             *  1009
* 53                          endif TYPE CODE IS INVALID                                        *  1010
* 54                      endif PRIORITY IS INVALID                                             *  1011
* 55                  endif DESTINATION IS INVALID                                              *  1012
* 56              endif STARTING LOCATION IS INVALID                                            *  1013
* 57          endif UNIT NAME IS NULL                                                           *  1014
* 58      enddo UNTIL UNIT←COUNTER EQUALS SIXTY                                                 *  1015
* 59  enddo UPON FUNCTION REQUEST                                                               *  1016
*                                                                                               *
*********************************************************************************************
```

ESTABLISH A FORWARD EDGE OF BATTLE AREA

```
REF
PAGE ***************************************************************************************
     *                                                                                     *
     *  1    .AN ARBITRARY MAXIMUM OF SIX NODES MAY BE DESIGNATED AS FEBA POINTS.         *   1018
     *  2    .                                                                            *   1019
     *  3    do UPON FUNCTION REQUEST                                                     *   1020
     *  4        do SIX TIMES                                                             *   1021
     *  5            PROMPT FOR MAP←NUMBER AND NODE←NUMBER, OR EXIT                       *   1022
     *  6            if INPUT IS EXIT                                                     *   1023
     *  7                return TO CALLING PROGRAM                                        *   1024
     *  8            endif INPUT IS EXIT                                                  *   1025
     *  9            CHECK MAP←NUMBER AND NODE←NUMBER FOR LEGITIMACY                      *   1026
     * 10            if NOT VALID                                                         *   1027
     * 11                PRINT ERROR MESSAGE                                              *   1028
     * 12                TRY AGAIN                                                        *   1029
     * 13            else VALID                                                           *   1030
     * 14                FIND NODE←VECTOR CORRESPONDING TO MAP←NUMBER, NODE←NUMBER        *   1031
     * 15                ASSIGN NODE←VECTOR TO FEBA ARRAY                                 *   1032
     * 16            endif NOT VALID                                                      *   1033
     * 17        enddo SIX TIMES                                                          *   1034
     * 18    enddo UPON FUNCTION REQUEST                                                  *   1035
     *                                                                                     *
     ***************************************************************************************
```

FIND NODE←VECTOR NEAREST TO UTM COORDINATES

```
REM
PROC ********************************************************************************
     *                                                                              *
     *  1    .THIS WILL BE ACCOMPLISHED BY STARTING AT THE BEGINNING OF THE         *  1037
     *  2    .NODE←VECTOR ARRAY AND COMPUTING EACH NODE←VECTOR 'S DISTANCE FROM THE *  1038
     *  3    .UTM COORDINATES USING A LEAST SQUARES APPROACH. THAT IS, DISTANCE IS  *  1039
     *  4    .CALCULATED TO BE THE SUM OF THE SQUARES OF THE DIFFERENCES BETWEEN THE*  1040
     *  5    .NODE←VECTOR 'S LATITUDE AND LONGITUDE AND THE CALCULATED UTM COORDI- *  1041
     *  6    .NATES.                                                                *  1042
     *  7    .                                                                      *  1043
     *  8    LEAST = INFINITY                                                       *  1044
     *  9    do FOR EACH NODE←VECTOR                                                *  1045
     * 10        DISTANCE = (UTM LAT. - LAT.) ** 2 + (UTM LON. - LON.) ** 2        *  1046
     * 11        if DISTANCE IS LESS THAN LEAST                                    *  1047
     * 12            REMEMBER THE NODE←VECTOR                                      *  1048
     * 13            LEAST = DISTANCE                                              *  1049
     * 14        endif DISTANCE IS LESS THAN LEAST                                 *  1050
     * 15   enddo FOR EACH NODE←VECTOR                                             *  1051
     * 16   .                                                                      *  1052
     * 17   .REMEMBERED NODE←VECTOR IS THE ONE WE WANT                             *  1053
     * 18   .                                                                      *  1054
     *                                                                              *
     ********************************************************************************
```

PATH DETERMINATION AND DISPLAY

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    .THESE ARE THE FUNCTIONS WHICH PERFORM THE FUNDAMENTAL MOVEMENT ALGO-   *  1056
     *  2    .RITHM. THE MAIN PURPOSE IS TO COMPUTE OPTIMAL PATHS FOR DEFINED        *  1057
     *  3    .GROUND FORCE UNITS, SO THAT CERTAIN CRITERIA IS SATISFIED. THE GENERAL *  1058
     *  4    .PROCEDURE IS, GIVEN A UNIT'S STARTING NODE, DESTINATION NODE, AND      *  1059
     *  5    .DESIRED ARRIVAL TIME AT DESTINATION, OR START TIME AT THE BEGINNING,   *  1060
     *  6    .CALCULATE THE BEST PATH FOR THE UNIT TO TRAVEL. 'BEST' IS USER-DEFINED *  1061
     *  7    .AS A FUNCTION OF TIME AND RISK. THE FORMULA USED IS K*TIME + C*RISK,   *  1062
     *  8    .WHERE K AND C ARE CONSTANTS, C EQUALING ZERO OR ONE, AND K BEING BE-   *  1063
     *  9    .TWEEN ZERO AND TEN. BOTH K AND C MAY NOT BE ZERO AT THE SAME TIME.     *  1064
     * 10    .USING THE BASIC PATH ALGORITHM, CERTAIN VARIATIONS OF THE PATH CALCU-  *  1065
     * 11    .LATION MAY BE PERFORMED, SUCH AS FINDING THE SECOND SHORTEST PATH,     *  1066
     * 12    .FINDING THE OPTIMAL NODE FOR INTERDICTION, AND, GIVEN A NUMBER OF UNITS*  1067
     * 13    .TRAVELING THROUGH A ROAD NETWORK, FIND WHICH ORDER OF PRIORITIES OPTI- *  1068
     * 14    .MIZES THE ENTIRE NETWORKS USAGE.                                       *  1069
     * 15    . THERE ARE TWO MORE OPTIONS FROM WHICH THE USER MAY CHOOSE; INTERDIC--  *  1070
     * 16    .TION AND SOLUTIONS. THEIR PROPERTIES WILL BE EXPLAINED LATER.          *  1071
     * 17    .                                                                       *  1072
     * 18    do UPON FUNCTION REQUEST                                                *  1073
     * 19        do FOREVER                                                          *  1074
     * 20            DISPLAY FUNCTION MENU                                           *  1075
     * 21            WAIT FOR USER INPUT                                             *  1076
     * 22            do CASE OF                                                      *  1077
     * 23 INTERDICTION:                                                              *  1078
  46 * 24                PERFORM INTERDICTIVE OPERATIONS                            *  1079
     * 25 PATH:                                                                      *  1080
  53 * 26                CALCULATE PATH(S) AND SOLUTION VECTORS                     *  1081
     * 27 SECOND:                                                                    *  1082
  60 * 28                CALCULATE SECOND BEST PATH                                 *  1083
     * 29 NETWORK:                                                                   *  1084
  61 * 30                FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL       *  1085
     * 31 BEST:                                                                      *  1086
  62 * 32                CALCULATE BEST NODE AT WHICH TO INTERDICT                  *  1087
```

```
    * 33 SOLUTION:                                                          *  1088
 63 * 34                 PRESENT RESULTS                                    *  1089
    * 35 EXIT:                                                             *  1090
    * 36                 return TO CALLING PROGRAM                          *  1091
    * 37              enddo CASE OF                                         *  1092
    * 38           enddo FOREVER                                            *  1093
    * 39       enddo UPON FUNCTION REQUEST                                  *  1094
    *                                                                       *
    *************************************************************************
```

PERFORM INTERDICTIVE OPERATIONS

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . HERE, THE USER MAY INTERDICT TO FORCE UNITS TO MOVE A CERTAIN        *     1096
     *  2    .WAY, TO MAKE LINKS IMPASSIBLE, OR TO SEE THE EFFECTS OF SUCH INTERDIC- *     1097
     *  3    .TION. THE USER MAY ALSO LOOK AT THE CONTENTS OF NODE←VECTORS.         *     1098
     *  4    .                                                                      *     1099
     *  5    do UPON FUNCTION REQUEST                                               *     1100
     *  6        do FOREVER                                                         *     1101
     *  7            DISPLAY FUNCTION MENU                                          *     1102
     *  8            WAIT FOR USER INPUT                                            *     1103
     *  9            do CASE OF                                                     *     1104
     * 10 VECTOR2:                                                                 *     1105
  47 * 11                PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINATES *     1106
     * 12 CONTENTS:                                                               *     1107
  48 * 13                CHANGE CONTENTS OF A NODE←VECTOR                       *     1108
     * 14 ERASE:                                                                  *     1109
  50 * 15                DELETE A NODE FROM THE NODE←VECTOR ARRAY                *     1110
     * 16 SHOW:                                                                   *     1111
  51 * 17                PRINT CONTENTS OF A NODE←VECTOR                        *     1112
     * 18 PROPERTIES:                                                             *     1113
  52 * 19                CHANGE PROPERTIES OF A UNIT                            *     1114
     * 20 EXIT:                                                                   *     1115
     * 21                return TO CALLING PROGRAM                              *     1116
     * 22            enddo CASE OF                                              *     1117
     * 23        enddo FOREVER                                                  *     1118
     * 24    enddo UPON FUNCTION REQUEST                                        *     1119
     *                                                                              *
     ********************************************************************************
```

PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINATES

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . THIS ALLOWS THE USER TO FIND OUT WHICH NODE←VECTOR IS NEAREST ANY UTM  *   1121
     *  2    .COORDINATES OF INTEREST. THIS IS OFTEN HELPFUL IF THE TRACKBALL IS      *   1122
     *  3    .UNAVAILABLE.                                                           *   1123
     *  4    .                                                                       *   1124
     *  5    PROMPT FOR USER INPUT UTM COORDINATES                                   *   1125
     *  6    SET LEAST TO INFINITY                                                   *   1126
     *  7    do FOR EACH NODE←VECTOR                                                 *   1127
     *  8        DISTANCE = (UTM LAT.-LATITUDE) ** 2 + (UTM LON. - LONGITUDE) **2    *   1128
     *  9        if DISTANCE IS LESS THAN LEAST                                      *   1129
     * 10            SET LEAST EQUAL TO DISTANCE                                     *   1130
     * 11            REMEMBER NODE←VECTOR                                            *   1131
     * 12        endif DISTANCE IS LESS THAN LEAST                                   *   1132
     * 13    enddo FOR EACH NODE←VECTOR                                              *   1133
     * 14    .                                                                       *   1134
     * 15    .THE ANSWER IS THE LAST REMEMBERED NODE←VECTOR                          *   1135
     * 16    .                                                                       *   1136
     * 17    PRINT OUT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR                        *   1137
     *                                                                              *
     ********************************************************************************
```

CHANGE CONTENTS OF A NODE←VECTOR

```
REF
PAGE ***********************************************************************
     *                                                                      *
     *  1    . THIS FUNCTION ALLOWS THE USER TO CHANGE CERTAIN CONDITIONS OF ANY                *  1139
     *  2    .NODE←VECTOR. THE USER MAY CHANGE ANY OF SIX LINK CHARACTERISTICS,                 *  1140
     *  3    .NUMBER OF LANES, LINK DISTANCE, TERRAIN CODE, BRIDGE CODE, CITY CODE,             *  1141
     *  4    .OR ROAD TYPE.                                                                     *  1142
     *  5    .                                                                                  *  1143
     *  6    do UPON FUNCTION REQUEST                                                           *  1144
     *  7        PROMPT ENTRY OF MAP←NUMBER, NODE←NUMBER                                        *  1145
     *  8        if ENTERED NODE←VECTOR IS NOT IN MEMORY                                        *  1146
     *  9            PRINT ERROR MESSAGE                                                        *  1147
     * 10            TRY AGAIN                                                                  *  1148
     * 11        else ENTERED NODE←VECTOR IS IN MEMORY                                          *  1149
     * 12            PROMPT FOR LINKNG MAP←NUMBER, NODE←NUMBER                                  *  1150
     * 13            if LINKING NODE←VECTOR IS NOT IN MEMORY                                    *  1151
     * 14                PRINT ERROR MESSAGE                                                    *  1152
     * 15                TRY AGAIN                                                              *  1153
     * 16            else LINKING NODE←VECTOR IS IN MEMORY                                      *  1154
     * 17                if LINKING NODE←VECTOR DOES NOT LINK TO FIRST ENTERED NODE←VECTOR       *  1155
     * 18                    PRINT ERROR MESSAGE                                                *  1156
     * 19                    TRY AGAIN                                                          *  1157
     * 20                else TWO NAMED NODE←VECTORS ARE IN MEMORY AND ARE LINKED               *  1158
     * 21                    PRINT LIST OF CHOICES FOR ALTERATION                               *  1159
     * 22                    WAIT FOR USER INPUT                                                *  1160
     * 23                    do CASE OF                                                         *  1161
     * 24 ROAD:                                                                                 *  1162
  49 * 25                        CHANGE CHARACTERISTIC ...ROAD TYPE                             *  1163
     * 26 LINK:                                                                                 *  1164
  49 * 27                        CHANGE CHARACTERISTIC ...LINK DISTANCE                         *  1165
     * 28 TERRAIN:                                                                              *  1166
  49 * 29                        CHANGE CHARACTERISTIC ...TERRAIN CODE                          *  1167
     * 30 LANES:                                                                                *  1168
  49 * 31                        CHANGE CHARACTERISTIC ...NUMBER OF LANES                       *  1169
     * 32 BRIDGE:                                                                             *  1170
```

```
49 * 33                    CHANGE CHARACTERISTIC ...BRIDGE CODE                                    *  1171
   * 34 CITY:                                                                                      *  1172
49 * 35                    CHANGE CHARACTERISTIC ...CITY CODE                                      *  1173
   * 36                  enddo CASE OF                                                            *  1174
   * 37                  return TO CALLING PROGRAM                                                *  1175
   * 38                endif LINKING NODE<VECTOR DOES NOT LINK TO FIRST ENTERED NODE<VECTOR       *  1176
   * 39              endif LINKING NODE<VECTOR IS NOT IN MEMORY                                   *  1177
   * 40            endif ENTERED NODE<VECTOR IS NOT IN MEMORY                                     *  1178
   * 41          enddo UPON FUNCTION REQUEST                                                      *  1179
   *                                                                                              *
   ************************************************************************************************
```

CHANGE CHARACTERISTIC

```
REF
PAGE ***************************************************************************
     *                                                                         *
     *  1    . THIS IS THE SAME PROCEDURE REGARDLESS OF WHICH CHARACTERISTIC IS BEING  *  1181
     *  2    .CHANGED.                                                          *  1182
     *  3    .                                                                  *  1183
     *  4    PRINT OUT OLD VALUE                                                *  1184
     *  5    PROMPT FOR NEW VALUE                                               *  1185
     *  6    if NEW VALUE WITHIN ACCEPTABLE LIMITS                              *  1186
     *  7        ASSIGN NEW VALUE TO NODE<VECTOR                                *  1187
     *  8    else NEW VALUE IS NOT ACCEPTABLE                                   *  1188
     *  9        PRINT ERROR MESSAGE                                            *  1189
     * 10        TRY AGAIN                                                      *  1190
     * 11    endif NEW VALUE IS WITHIN ACCEPTABLE LIMITS                        *  1191
     *                                                                         *
     ***************************************************************************
```

DELETE A NODE FROM THE NODE←VECTOR ARRAY

```
REF
PAGE ***************************************************************************************************
     *                                                                                                   *
     *  1    . FOR THE PURPOSE OF INTERDICTION IT MAY BE DESIRED THAT A NODE←VECTOR                      *    1193
     *  2    .BE 'ERASED', THAT IS, AFFECTED SO THAT NO UNIT WILL TRAVEL THROUGH IT.                     *    1194
     *  3    .THIS ROUTINE CHANGES THE NUMBER OF LANES PARAMETER FOR EACH ADJACENT                       *    1195
     *  4    .LINK TO 0. IF THERE'S NO ROAD, NO UNIT CAN TRAVEL THERE. ONCE                              *    1196
     *  5    .DONE, RESTORATION IS PERFORMED BY EITHER REESTABLISHING EACH LINK                          *    1197
     *  6    .MANUALLY, OR BY READING THE MAPS FROM DISK AGAIN.                                          *    1198
     *  7    .                                                                                           *    1199
     *  8    do UPON FUNCTION REQUEST                                                                    *    1200
     *  9        PROMPT FOR MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR TO BE ERASED                          *    1201
     * 10        if ENTERED VALUES ARE INVALID                                                           *    1202
     * 11            PRINT ERROR MESSAGE                                                                 *    1203
     * 12            TRY AGAIN                                                                           *    1204
     * 13        else ENTERED VALUES ARE VALID                                                           *    1205
     * 14            do FOR EACH ADJACENT NODE                                                           *    1206
     * 15                SET NUMBER OF LANES TO THIS NODE TO ZERO                                        *    1207
     * 16            enddo FOR EACH ADJACENT NODE                                                        *    1208
     * 17            return TO CALLING PROGRAM                                                           *    1209
     * 18        endif ENTERED VALUES ARE INVALID                                                        *    1210
     * 19    enddo UPON FUNCTION REQUEST                                                                 *    1211
     *                                                                                                   *
     ***************************************************************************************************
```

PRINT CONTENTS OF A NODE←VECTOR

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    do UPON FUNCTION REQUEST                                               *  1213
     *  2       PROMPT FOR MAP←NUMBER, NODE←NUMBER                                  *  1214
     *  3       if ENTERED VALUES ARE NOT VALID                                     *  1215
     *  4          PRINT ERROR MESSAGE                                              *  1216
     *  5          TRY AGAIN                                                        *  1217
     *  6       else ENTERED VALUES ARE VALID                                       *  1218
     *  7          PRINT CONTENTS OF NODE←VECTOR                                    *  1219
     *  8          return TO CALLING PROGRAM                                        *  1220
     *  9       endif ENTERED VALUES ARE NOT VALID                                  *  1221
     * 10    enddo UPON FUNCTION REQUEST                                            *  1222
     *                                                                              *
     ********************************************************************************
```

CHANGE PROPERTIES OF A UNIT

```
REF
PAGE ************************************************************************************
     *                                                                                  *
     *  1    . AT THIS TIME, THE ONLY PROPERTY WHICH WE WILL ALLOW TO BE CHANGED,       *     1224
     *  2    .IS THE UNIT'S PRIORITY.                                                   *     1225
     *  3    .                                                                          *     1226
     *  4    do UPON FUNCTION REQUEST                                                   *     1227
     *  5        PROMPT ENTRY OF UNIT NAME                                              *     1228
     *  6        if INPUT NAME HAS AN ENTRY IN UNITS FILE                               *     1229
     *  7            PRINT OLD VALUE                                                    *     1230
     *  8            PROMPT ENTRY OF NEW PRIORITY                                       *     1231
     *  9            if INPUT IS BETWEEN 0 AND 127                                      *     1232
     * 10                SET NEW VALUE IN UNIT<VECTOR                                   *     1233
     * 11                return TO CALLING PROGRAM                                      *     1234
     * 12            else INPUT IS NOT VALID                                            *     1235
     * 13                PRINT ERROR MESSAGE                                            *     1236
     * 14                TRY AGAIN                                                      *     1237
     * 15            endif INPUT IS BETWEEN 0 AND 127                                   *     1238
     * 16        else INPUT NAME HAS NO ENTRY IN UNITS FILE                             *     1239
     * 17            PRINT ERROR MESSAGE                                                *     1240
     * 18            TRY AGAIN                                                          *     1241
     * 19        endif INPUT NAME HAS AN ENTRY IN UNITS FILE                            *     1242
     * 20   enddo UPON FUNCTION REQUEST                                                 *     1243
     *                                                                                  *
     ************************************************************************************
```

CALCULATE PATH(S) AND SOLUTION VECTORS

```
REF
PAGE ***************************************************************************************
     *                                                                                     *
     *  1    . THIS ROUTINE WILL DETERMINE THE BEST PATH FOR GROUND FORCE                  *  1245
     *  2    .UNITS TO USE, TAKING INTO CONSIDERATION TRAVEL TIME, GROUND COVER, TER-      *  1246
     *  3    .RAIN, BRIDGES, AND CITIES.                                                   *  1247
     *  4    . BEST DOESN'T ONLY IMPLY FASTEST, BUT MAY MEAN LEAST RISKY, OR SOME          *  1248
     *  5    .COMBINATION OF SPEED AND RISK.                                               *  1249
     *  6    .THERE ARE SEVERAL OPTIONS FROM WHICH THE USER IS FREE TO CHOOSE. AMONG       *  1250
     *  7    .THESE ARE:                                                                   *  1251
     *  8    . 1) SIMPLE BEST PATH CALCULATION, REGARDLESS OF TIME CONFLICTS,              *  1252
     *  9    . 2) AUTOMATIC CONFLICT RESOLUTION; THE PROGRAM DOES THE RESOLUTION           *  1253
     *  10   .WITHOUT USER INTERVENTION.                                                   *  1254
     *  11   . 3) WORK FORWARDS IN TIME, THAT IS, GIVE A STARTING TIME AND NODE,           *  1255
     *  12   .AND A DESTINATION NODE. THE PROGRAM WILL THEN WORK FORWARD IN TIME,          *  1256
     *  13   .BEGINNING AT THE START NODE, AND ENDING AT THE DESTINATION NODE.            *  1257
     *  14   . 4) WORK BACKWARDS IN TIME. GIVEN A STARTING NODE, DESTINATION NODE         *  1258
     *  15   .AND ARRIVAL TIME, START AT THE DESTINATION NODE, WORK BACKWARDS IN TIME     *  1259
     *  16   .TO THE STARTING NODE.                                                       *  1260
     *  17   . AFTER THE PATH HAS BEEN DETERMINED, THE ROUTE IS RECONSTRUCTED USING       *  1261
     *  18   .SOLUTION-VECTORS. THESE VECTORS RECORD THE TIMES DURING WHICH THE VAR-      *  1262
     *  19   .IOUS NODES ARE BUSY. THIS INFORMATION WILL BE USED WHEN RESOLVING           *  1263
     *  20   .TIME CONFLICTS AMONG THE PATHS.                                             *  1264
     *  21   .                                                                            *  1265
     *  22   do UPON FUNCTION REQUEST                                                     *  1266
     *  23       .                                                                        *  1267
     *  24       .BEFORE ANYTHING ELSE, CLEAR AWAY ALL OLD ROUTES AND SOLUTIONS           *  1268
     *  25       .                                                                        *  1269
  54 *  26       PURGE ALL ROUTES                                                         *  1270
     *  27       .                                                                        *  1271
     *  28       .NOW GET PATH PARAMETERS                                                 *  1272
     *  29       .                                                                        *  1273
     *  30       PROMPT FOR ENTRY OF MOVEMENT TYPE (BACKWARD OR FORWARD IN TIME)          *  1274
     *  31       PROMPT FOR ENTRY OF TIME CONFLICT RESOLUTION CODE (NONE OR AUTOMATIC)    *  1275
     *  32       .                                                                        *  1276
```

```
* 33        .HAVE USER ERTER C AND K, THE COEFFICIENTS FOR RISK AND TIME                    *  1277
* 34        .                                                                               *  1278
* 35        PROMPT FOR ENTRY OF C                                                           *  1279
* 36        if C NOT EQUAL TO 0 OR 1                                                        *  1280
* 37            PRINT MESSAGE                                                               *  1281
* 38            TRY AGAIN                                                                   *  1282
* 39        endif C NOT EQUAL TO 0 OR 1                                                     *  1283
* 40        PROMPT FOR ENTRY OF K                                                           *  1284
* 41        if K NOT BETWEEN 0 AND 9.99                                                     *  1285
* 42            PRINT ERROR MESSAGE                                                         *  1286
* 43            TRY AGAIN                                                                   *  1287
* 44        endif K NOT BETWEEN 0 AND 9.99                                                  *  1288
* 45        if C AND K EQUAL 0                                                              *  1289
* 46            PRINT ERROR MESSAGE                                                         *  1290
* 47            TRY AGAIN                                                                   *  1291
* 48        endif C AND K EQUAL 0                                                           *  1292
* 49        .                                                                               *  1293
* 50        .PROMPT FOR UNIT NAMES TO BE CONSIDERED                                         *  1294
* 51        .                                                                               *  1295
* 52        do UNTIL USER ENTERS NULL                                                       *  1296
* 53            PROMPT FOR ENTRY OF UNIT NAME                                               *  1297
* 54            if INPUT IS NULL                                                            *  1298
* 55                undo UNTIL USER ENTERS NULL                                             *  1299
* 56            else INPUT IS NOT NULL                                                      *  1300
* 57                REMEMBER UNIT                                                           *  1301
* 58            endif INPUT IS NULL                                                         *  1302
* 59        enddo UNTIL USER ENTERS NULL                                                    *  1303
* 60        .                                                                               *  1304
* 61        .CALCULATE THE PATHS                                                            *  1305
* 62        .                                                                               *  1306
* 63        do FOR EACH ENTERED UNIT, IN ORDER OF PRIORITY                                  *  1307
* 64            if MOVEMENT TYPE IS BACKWARDS                                               *  1308
* 65                PROMPT FOR ARRIVAL TIME                                                 *  1309
* 66            else MOVEMENT TYPE IS FORWARDS                                              *  1310
* 67                PROMPT FOR LEAVING TIME                                                 *  1311
* 68            endif MOVEMENT TYPE IS BACKWARDS                                            *  1312
* 69            .                                                                           *  1313
* 70            .THE 'WORKING LIST' IS A FICTITIOUS ARRAY WHICH KEEPS TRACK OF VARIOUS      *  1314
```

```
    * 71          .QUANTITIES NECESSARY FOR THE PATH ALGORITHM. THESE ARE WORTH MEASURE,          *    1315
    * 72          .PREDECESSOR NODE NUMBER, CUMMULATIVE TIME, CUMMULATIVE WORTH MEASURE,          *    1316
    * 73          .TIME MEASURE, SOLUTION←VECTOR POINTER, LANE USED, RISK MEASURE, AND            *    1317
    * 74          .PARK TIME. THESE VALUES ARE HELD IN THE NODE←VECTOR FOR EASY REFER-            *    1318
    * 75          .ENCE.                                                                          *    1319
    * 76          .                                                                               *    1320
    * 77          if TIME CONFLICT RESOLUTION IS AUTOMATIC                                        *    1321
 55 * 78              REINITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES                             *    1322
    * 79              SET K TO FIRST NODE TO BE CONSIDERED (DEPENDENT ON MOVEMENT TYPE)           *    1323
    * 80          else TIME CONLICT RESOLUTION IS NOT AUTOMATIC                                   *    1324
    * 81              if THIS UNIT'S MOVEMENT TYPE, ENTERED TIME, AND UNIT TYPE DIFFERS FROM/     *    1325
    * 82                  THE PREVIOUS UNIT'S                                                     *    1326
 55 * 83                  REINITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES                         *    1327
    * 84                  SET K TO FIRST NODE TO BE CONSIDERED (DEPENDENT ON MOVEMENT TYPE)       *    1328
    * 85              endif THIS UNIT'S PARAMETERS DIFFER                                         *    1329
    * 86          endif TIME CONFLICT RESOLUTION IS AUTOMATIC                                     *    1330
    * 87          .                                                                               *    1331
    * 88          .DURING THIS ALGORITHM, K IS THE NODE FROM WHICH THE ALGORITHM STARTS.          *    1332
    * 89          .THIS IS DETERMINED BY THE MOVEMENT TYPE TO BE EITHER THE START NODE            *    1333
    * 90          .(TYPE IS FORWARD) OR THE DESTINATION NODE (TYPE IS BACKWARD). I IS THE         *    1334
    * 91          .FINAL NODE, THE OPPOSITE OF K (IF K IS THE START NODE, I IS THE DESTI-         *    1335
    * 92          .NATION, AND VICE VERSA). J IS THE NODE UNDER CURRENT CONSIDERATION BY          *    1336
    * 93          .THE ALGORITHM.                                                                 *    1337
    * 94          .                                                                               *    1338
    * 95          SET I TO END NODE (DEPENDENT UPON MOVEMENT TYPE)                                *    1339
    * 96          SET J TO K                                                                      *    1340
    * 97          do UNTIL NODE I IS LABELED ... TEST IMMEDIATELY FOR EARLY EXIT                  *    1341
    * 98              if I IS LABELED                                                             *    1342
    * 99                  undo UNTIL NODE I IS LABELED                                            *    1343
    * 00              endif I IS LABELED                                                          *    1344
    * 01              if J ISN'T LABELED                                                          *    1345
    * 02                  LABEL NODE J                                                            *    1346
    * 03                  SET J'S CUMMULATIVE TIME                                                *    1347
    * 04                  do FOR EACH OF J'S ADJACENT NODES                                       *    1348
    * 05                      if IT IS LABELED                                                    *    1349
    * 06                          cycle FOR EACH OF J'S ADJACENT NODES                            *    1350
    * 07                      else IT IS NOT LABELED                                              *    1351
 56 * 08                          CALCULATE AND ASSIGN TIME AND WORTH VALUES                      *    1352
```

```
     * 09                    endif IT IS LABELED                                    *    1353
     * 10                 enddo FOR EACH OF J'S ADJACENT NODES                      *    1354
     * 11              endif J ISN'T LABELED                                        *    1355
  57 * 12              COMPUTE NEXT NODE TO LABEL ...DEPENDENT UPON TYPE            *    1356
     * 13              SET J TO NEW NODE                                            *    1357
     * 14           enddo UNTIL NODE I LABELED                                      *    1358
  59 * 15           COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR                       *    1359
     * 16        enddo FOR EACH UNIT                                               *    1360
     * 17        return TO CALLING PROGRAM                                         *    1361
     * 18     enddo UPON FUNCTION REQUEST                                          *    1362
     *                                                                             *
     ***********************************************************************************
****** TOO MANY LINES IN SEGMENT
```

```
     PURGE ALL ROUTES


REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . THIS ROUTINE WILL GET RID OF ALL PREVIOUS PATHS WHICH HAVE BEEN      *   1364
     *  2    .CALCULATED. THIS CONSISTS OF ERASING ALL SOLUTION<VECTORS AND         *   1365
     *  3    .ROUTE<VECTORS, RESETING THEIR COUNTERS TO ZERO, AND ERASING ALL POINT-*   1366
     *  4    .ERS WHICH REFERENCE THESE ARRAYS.                                     *   1367
     *  5    .                                                                      *   1368
     *  6    do FOR EACH ROUTE<VECTOR                                               *   1369
     *  7        SET CONTENTS TO 0                                                  *   1370
     *  8    enddo FOR EACH ROUTE<VECTOR                                            *   1371
     *  9    do FOR EACH SOLUTION<VECTOR                                            *   1372
     * 10        SET CONTENTS TO 0                                                  *   1373
     * 11    enddo FOR EACH SOLUTION<VECTOR                                         *   1374
     * 12    SET COUNTERS TO 0                                                      *   1375
     * 13    do FOR EACH NODE<VECTOR                                                *   1376
     * 14        SET SOLUTION<VECTOR POINTER TO 0                                   *   1377
     * 15    enddo FOR EACH NODE<VECTOR                                             *   1378
     *                                                                              *
     ********************************************************************************
```

REINITIALIZE "WORKING LIST" NODE←VECTOR ENTRIES

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . THIS FUNCTION MUST BE PERFORMED EACH TIME WE WISH TO                 *     1380
     *  2    ."WIPE THE SLATE CLEAN" AND START A NEW PATH-FINDING PROCEDURE.        *     1381
     *  3    .                                                                      *     1382
     *  4    do FOR EACH NODE←VECTOR                                                *     1383
     *  5         SET TIME MEASURE TO ZERO                                          *     1384
     *  6         SET WORTH MEASURE TO ZERO                                         *     1385
     *  7         SET CUMMULATIVE TIME TO ZERO                                      *     1386
     *  8         SET CUMMULATIVE WORTH TO 32767                                    *     1387
     *  9         SET LABEL TO ZERO                                                 *     1388
     * 10         SET PREDECESSOR TO -1                                             *     1389
     * 11         SET LANE TO ONE                                                   *     1390
     * 12         SET PARKING TIME TO ZERO                                          *     1391
     * 13         SET RISK MEASURE TO 0                                             *     1392
     * 14    enddo FOR EACH NODE←VECTOR                                             *     1393
     *                                                                              *
     ********************************************************************************
```

CALCULATE AND ASSIGN TIME AND WORTH VALUES

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . WHEN A NODE IS FOUND BY THE PATH ALGORITHM TO BE THE NEXT TO BE       *  1395
     *  2    .LABELED, THE TIME AND WORTH VALUES OF ALL OF ITS LINKS TO UNLABELED    *  1396
     *  3    .ADJACENT NODES ARE CALCULATED AND ASSIGNED. THAT PROCESS OF CALCULAT-  *  1397
     *  4    .ING AND ASSIGNING IS DONE BY THIS ROUTINE.                             *  1398
     *  5    . THERE ARE TWO TYPES OF TIME AND WORTH: MEASURE AND CUMMULATIVE. TIME  *  1399
     *  6    .MEASURE AND WORTH MEASURE REFER TO THE VALUE CALCULATED FOR THE LINK   *  1400
     *  7    .BETWEEN A NODE AND ONE OF ITS UNLABELED ADJACENT NODES. CUMMULATIVE    *  1401
     *  8    .TIME AND CUMMULATIVE WORTH REFER TO THE ACCUMULATED TIME AND WORTH     *  1402
     *  9    .WHICH HAS BUILT UP DURING THE PATH CALCULATION. THUS CUMMULATIVE TIME  *  1403
     * 10    .IS THE TIME OF DAY AT WHICH A UNIT ARRIVES AT THE ADJACENT NODE, AND   *  1404
     * 11    .CUMMULATIVE WORTH IS THE WORTH OF THE PATH AT THAT POINT IN TIME.      *  1405
     * 12    . WORTH IS A COMBINATION OF RISK AND TIME, USING PARAMETERS SPECIFIED   *  1406
     * 13    .BY THE USER. THE EQUATION IS C * RISK + K * TIME, WHERE C MAY BE       *  1407
     * 14    .0 OR 1, AND K MAY BE BETWEEN 0 AND 9.99, BUT BOTH MAY NOT EQUAL 0      *  1408
     * 15    .AT THE SAME TIME. A BEST PATH IS A PATH WHICH MINIMIZES WORTH. BY       *  1409
     * 16    .ADJUSTING THE PARAMETERS C AND K, THE USER MAKES HIS OWN DEFINITION    *  1410
     * 17    .OF BEST. IT COULD BE THAT BEST IS THE LEAST RISKY (C=1,K=0), FASTEST   *  1411
     * 18    .(C=0,K=1), OR A COMBINATION OF THE TWO.                                *  1412
     * 19    . RISK IS CALCULATED BY SUMMING THE NUMERICAL VALUES ASSIGNED TO THE    *  1413
     * 20    .FOUR RISK FACTORS: GROUND COVER, TERRAIN, BRIDGE, AND CITY CODES.      *  1414
     * 21    .                                                                       *  1415
     * 22    CALCULATE LINK TRAVERSAL TIME                                           *  1416
     * 23    if MOVEMENT TYPE IS FORWARD                                             *  1417
     * 24        CUMMULATIVE TIME IS PREDECESSOR'S CUMMULATIVE TIME + LINK TIME      *  1418
     * 25    else MOVEMENT TYPE IS BACKWARD                                          *  1419
     * 26        CUMMULATIVE TIME IS THIS NODE'S CUMMULATIVE TIME - LINK TIME        *  1420
     * 27    endif MOVEMENT TIME IS FORWARD                                          *  1421
     * 28    .                                                                       *  1422
     * 29    .RESOLVE TIME CONFLICTS                                                 *  1423
     * 30    .                                                                       *  1424
     * 31    if USER DOES NOT WANT TIME CONFLICTS RESOLVED                           *  1425
     * 32        CONTINUE                                                            *  1426
```

```
* 33    elseif THIS NODE←VECTOR HAS NEVER BEEN USED BY A PATH                    *  1427
* 34        CONTINUE                                                             *  1428
* 35    else THIS NODE←VECTOR HAS BEEN USED BY A PATH                            *  1429
* 36        .                                                                    *  1430
* 37        .LOOK FOR TIME CONFLICT. IF ONE IS FOUND, RESOLVE IT                 *  1431
* 38        .                                                                    *  1432
* 39        SET ALL FOUR LANES TO CUMMULATIVE TIME                               *  1433
* 40        do FOR EACH SOLUTION←VECTOR REFERING TO THIS NODE                    *  1434
* 41            if THE TIME INTERVAL SHOULD NOT BE CONSIDERED                    *  1435
* 42                cycle FOR EACH SOLUTION←VECTOR                               *  1436
* 43            else THE TIME INTERVAL SHOULD BE CONSIDERED                      *  1437
* 44                SET INDEX TO SOLUTION←VECTOR 'S LANE NUMBER                  *  1438
* 45                if THIS UNIT CAN CLEAR THIS NODE BEFORE THE TIME INTERVAL    *  1439
* 46                    cycle FOR EACH SOLUTION←VECTOR                           *  1440
* 47                else THIS UNIT CANNOT CLEAR THIS NODE BEFORE THE TIME INTERVAL *  1441
* 48                    if THIS UNIT DOES NOT ARRIVE UNTIL AFTER THE TIME INTERVAL *  1442
* 49                        cycle FOR EACH SOLUTION←VECTOR                       *  1443
* 50                    else THIS UNIT ARRIVES DURING THE TIME INTERVAL          *  1444
* 51                        .                                                    *  1445
* 52                        .THERE IS A TIME CONFLICT                            *  1446
* 53                        .                                                    *  1447
* 54                        if MOVEMENT TYPE IS FORWARD                          *  1448
* 55                            HOLD UP UNIT UNTIL NODE CLEARS                   *  1449
* 56                            RECORD THIS TIME IN LANE (INDEX)                 *  1450
* 57                        else MOVEMENT TYPE IS BACKWARD                       *  1451
* 58                            HAVE UNIT ARRIVE BEFORE TIME INTERVAL            *  1452
* 59                            RECORD THIS TIME IN LANE (INDEX)                 *  1453
* 60                        endif MOVEMENT TYPE IS FORWARD                       *  1454
* 61                        .                                                    *  1455
* 62                        .THAT'S ALL FOR THIS SOLUTION←VECTOR                 *  1456
* 63                        .                                                    *  1457
* 64                    endif THIS UNIT DOES NOT ARRIVE UNTIL AFTER THE TIME INTERVAL *  1458
* 65                endif THIS UNIT CAN CLEAR THIS NODE BEFORE THE TIME INTERVAL *  1459
* 66            endif THE TIME INTERVAL SOULD NOT BE CONSIDERED                  *  1460
* 67        enddo FOR EACH SOLUTION←VECTOR REFERING TO THIS NODE                 *  1461
* 68        .                                                                    *  1462
* 69        .FIND THE BEST LANE (FOR FORWARD, BEST IS LEAST TIME, FOR BACKWARD, BEST *  1463
* 70        .IS THE GREATEST TIME).                                              *  1464
```

```
* 71        .                                                              *  1465
* 72        if MOVEMENT IS FORWARD                                         *  1466
* 73            SET BEST TO 32767                                          *  1467
* 74        else MOVEMENT IS BACKWARD                                      *  1468
* 75            SET BEST TO 0                                              *  1469
* 76        endif MOVEMENT IS FORWARD                                      *  1470
* 77        do FOR EACH AVAILABLE LANE FOR THIS LINK                       *  1471
* 78            if TIME OF LANE IS LESS (GREATER) THAN BEST                *  1472
* 79                SET BEST TO TIME OF LANE                               *  1473
* 80                REMEMBER LANE NUMBER                                   *  1474
* 81            endif TIME OF LANE IS LESS (GREATER) THAN BEST             *  1475
* 82        enddo FOR EACH AVAILABLE LANE FOR THIS LINK                    *  1476
* 83        .                                                              *  1477
* 84        .ANY TIME CONFLICT HAS NOW BEEN RESOLVED. THE BEST RESULT FROM THE  *  1478
* 85        .PREVIOUS SECTION IS NOW THE 'TIME OF RECORD'.                 *  1479
* 86        .                                                              *  1480
* 87        CALCUATE PARKING TIME ('TIME OF RECORD' LESS CUMMULATIVE TIME)  *  1481
* 88        .                                                              *  1482
* 89        .PARKING TIME AFFECTS THE ENTIRE LENGTH OF THE UNIT, SO IT MUST BE  *  1483
* 90        .SEEN IF THE ADITION OF THIS PARKING TIME CREATES ANY NEW CONFLICTS.  *  1484
* 91        .IF SO, THIS LINK CANNOT BE USED.                              *  1485
* 92        .                                                              *  1486
* 93        if PARKING TIME EQUALS ZERO                                    *  1487
* 94            CONTINUE                                                   *  1488
* 95        else PARKING TIME DOES NOT EQUAL ZERO                          *  1489
* 96            if PARKING TIME CREATES NEW CONFLICTS                      *  1490
* 97                MAKE NO ASSIGNMENTS OF TIME AND WORTH                  *  1491
* 98                return                                                 *  1492
* 99            endif PARKING TIME CREATES NEW CONFLICTS                   *  1493
* 00        endif PARKING TIME EQUALS ZERO                                 *  1494
* 01    endif THE USER DOES NOT WANT TIME CONFLICTS RESOLVED               *  1495
* 02    .                                                                  *  1496
* 03    .NOW CALCULATE RISK AND MAKE ASSIGNMENTS                           *  1497
* 04    .                                                                  *  1498
* 05    CALCULATE RISK                                                     *  1499
* 06    CALCULATE WORTH FOR THIS LINK                                      *  1500
* 07    CALCULATE CUMMULATIVE WORTH                                        *  1501
* 08    if CUMMULATIVE WORTH IS LESS THAN CURRENT CUMMULATIVE WORTH        *  1502
```

```
     * 09        MAKE ASSIGNMENTS OF TIME, WORTH, PARKING, AND LANE                    *    1503
     * 10     endif CUMMULATIVE WORTH IS LESS THAN CURRENT CUMMULATIVE WORTH           *    1504
     *                                                                                 *
     ***********************************************************************************
***** TOO MANY LINES IN SEGMENT
```

COMPUTE NEXT NODE TO LABEL

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    .                                                                      *  1506
     *  2    . THIS CONSISTS OF FINDING WHICH UNLABELED NODE VECTOR HAS THE          *  1507
     *  3    .LEAST CUMMULATIVE WORTH. BY MAKING WORTH A FUNCTION SUCH THAT         *  1508
     *  4    .THE BETTER A NODE'S WORTH,THE SMALLER THE WORTH VALUE, CUM-           *  1509
     *  5    .MULATIVE WORTH MUST BE ORIGINALLY SET TO INFINITY, OR IN THIS         *  1510
     *  6    .CASE, 32767.                                                          *  1511
     *  7    .                                                                      *  1512
     *  8    SET LEAST TO 32767                                                     *  1513
     *  9    do FOR EACH NODE                                                       *  1514
     * 10        if NODE IS LABELED                                                 *  1515
     * 11            cycle                                                          *  1516
     * 12        elseif WORTH MEASURE < LEAST                                       *  1517
     * 13            SET LEAST TO WORTH MEASURE                                     *  1518
     * 14            SAVE NODE NUMBER                                               *  1519
     * 15        endif NODE IS LABELED                                              *  1520
     * 16   enddo FOR EACH NODE                                                     *  1521
     *                                                                              *
     ********************************************************************************
```

EXPLANATION OF PATH DETERMINATION ALGORITHM

```
#                                                                                #
#  1        OUR PROBLEM IS TO DETERMINE A ROUTE FOR A GROUND FORCE UNIT TO       #    1523
#  2    FOLLOW, SUCH THAT THE PATH GETS THE UNIT TO ITS DESTINATION NODE FROM    #    1524
#  3    ITS STARTING NODE AS QUICKLY AND/OR AS LEAST RISKY AS POSSIBLE.  THIS    #    1525
#  4    IS A SHORTEST PATH PROBLEM, WITH SOME TWISTS THROWN IN.  ONE OF THESE    #    1526
#  5    TWISTS IS THAT EACH UNIT HAS A DIFFERENT RATE OF TRAVEL, WITH THAT RATE  #    1527
#  6    VARYING ACCORDING TO THE TIME OF DAY.  ANOTHER IS THAT WE REALLY HAVE    #    1528
#  7    MORE THAN ONE UNIT, AND WHEN MORE THAN ONE UNIT HAS THE SAME DESTINATION #    1529
#  8    NODE, WE MAY WISH FOR THE UNITS TO ARRIVE THERE AT THE SAME TIME.  WE HAVE #    1530
#  9    FOUND THAT THE SIMULTANEOUS DESTINATION TIME PROBLEM MAY BE SOLVED BY STARTING #    1531
# 10    AT THE DESTINATION NODE AND WORKING OUR WAY BACK TO THE STARTING NODE, RUN- #    1532
# 11    NING TIME IN REVERSE.                                                    #    1533
# 12        PERHAPS IT WOULD BE BEST TO FIRST EXPLAIN DIJKSTRA'S ALGORITHM FOR ONE UNIT #    1534
# 13    SUPPOSE THAT THERE ARE N NODES, NUMBERED FROM 1 TO N.  AND SUPPOSE FURTHER #    1535
# 14    THAT WE WISH TO FIND THE SHORTEST PATH BETWEEN NODES 1 AND N.  THAT IS, NODE 1 #    1536
# 15    IS THE DESTINATION, AND NODE N IS THE STARTING NODE.  LABEL 1 WITH THE PERM- #    1537
# 16    ANENT VALUE ZERO, AND TENTATIVELY LABEL ALL OTHERS WITH VALUE INFINITY.  EACH #    1538
# 17    TIME A PERMANENT LABEL IS ASSIGNED, IT REPRESENTS THE SHORTEST DISTANCE BETWEEN #    1539
# 18    THAT NODE AND THE DESTINATION NODE, NODE N.                              #    1540
# 19        ONE BY ONE, COMPARE EACH NODE LABEL EXCEPT THAT AT 1 WITH THE SUM OF THE #    1541
# 20    LABEL OF NODE 1 (THAT IS, 0 ) AND THE DIRECT DISTANCE FROM NODE 1 TO THE NODE #    1542
# 21    IN QUESTION.  THE SMALLER OF THE TWO NUMBERS IS THE NEW TENTATIVE LABEL. #    1543
# 22        NEXT, DETERMINE THE SMALLEST OF THE N-1 TENTATIVE LABELS AND DECLARE IT #    1544
# 23    PERMANENT.  SUPPOSE THAT NODE K IS THE ONE PERMANENTLY LABELED.  THEN,   #    1545
# 24    ONE AT A TIME, COMPARE EACH OF THE N-2 REMAINING TENTATIVE NODE LABELS TO #    1546
# 25    THE SUM OF THE LABEL JUST ASSIGNED PERMANENTLY TO NODE K AND THE DIRECT  #    1547
# 26    DISTANCE FROM NODE K TO THE NODE UNDER CONSIDERATION.  THE SMALLER OF THE TWO #    1548
# 27    NUMBERS BECOMES THE TENTATIVE LABEL.  DETERMINE THE MINIMUM OF THE N-2 TEN- #    1549
# 28    TATIVE LABELS, DECLARE IT PERMANENT, AND MAKE IT THE BASIS OF ANOTHER MOD- #    1550
# 29    IFICATION OF THE REMAINING TENTATIVE LABELS OF THE TYPE DESCRIBED ABOVE. #    1551
# 30    WHEN, AFTER AT MOST N-1 EXECUTIONS OF THE FUNDAMENTAL ITERATIVE STEP, NODE #    1552
# 31    N IS PERMANENTLY LABELED, THE PROCEDURE TERMINATES.                      #    1553
# 32        THE PATH FROM N TO 1 MAY BE RECONSTRUCTED IF, FOR EACH LABELED NODE, THE #    1554
# 33    NODE FROM WHICH IT WAS LABELED (ITS PREDECESSOR) IS RECORDED.  THEN, BY  START- #    1555
# 34    ING AT N, WE MAY SIMPLY FOLLOW THE PREDECESSORS BACK TO THE DESTINATION NODE. #    1556
# 35        THIS PROCESS IS CORRECT, BUT FOR EASE OF HANDLING ON THE COMPUTER, CERTAIN #    1557
# 36    ADJUSTMENTS HAVE BEEN MADE.  THESE WERE ORIGINALLY SUGGESTED BY MINTY AND OTHERS #    1558
```

```
# 37    INSTEAD OF CONSIDERING A LABEL TO POSSESS A VALUE, WE SAY THAT A NODE IS EITHER          #    1559
# 38    LABELED OR IT IS NOT, AND WE CONSIDER THE DISTANCE FROM NODE 1 TO BE A SEPARATE          #    1560
# 39    ENTITY.  AND INSTEAD OF USING INFINITY, WE USE THE LARGEST NUMBER ALLOWABLE.             #    1561
# 40         WE ALSO DO SOMETHING WHICH WASN'T ANTICIPATED BY THESE GENTLEMEN, AND THAT          #    1562
# 41    IS TO SOMETIMES RUN THE ALGORITHM IN REVERSE.  NOW THE BASIC STEPS ARE STILL             #    1563
# 42    THE SAME EXCEPT THAT WE ARE RUNNING TIME IN REVERSE, USING AN ARRIVAL TIME               #    1564
# 43    AS THE STARTING POINT.  THUS, ALL DISTANCES ARE NEGATIVE, AND INSTEAD OF                 #    1565
# 44    LOOKING FOR THE LEAST 'LABEL', WE LOOK FOR THE GREATEST.  BY DOING THIS,                 #    1566
# 45    WE FIND OUT AT WHAT TIME A UNIT HAD TO LEAVE ITS STARTING NODE IN ORDER TO               #    1567
# 46    ARRIVE AT ITS DESTINATION NODE AT THE GIVEN TIME.                                        #    1568
# 47    THUS WE HAVE TWO CHOICES IN APPROACH TO THIS PROBLEM.  WE MAY WISH TO FIND               #    1569
# 48    THE SHORTEST PATH FROM NODE 1 TO NODE N, OR WE MAY FIND WHICH NODE IS SO FAR             #    1570
# 49    FROM NODE N.                                                                             #    1571
# 50         RATHER THAN WORK WITH STRICTLY DISTANCE, WE WILL REALLY BE WORKING WITH          #    1572
# 51    TIME, AND THE TWO QUESTINS WE WILL ADDRESS WILL BE 1) HOW FAST CAN WE GET FROM           #    1573
# 52    NODE 1 TO NODE N? AND 2) GIVEN A TIME WE WISH TO ARRIVE AT NODE N, AT WHAT TIME          #    1574
# 53    SHOULD WE LEAVE NODE 1?                                                                  #    1575
#                                                                                              #
################################################################################################
```

COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . SOLUTION←VECTORS ARE CONSTRUCTED BY TRACING THE PATH OF NODE←VECTORS.  *  1577
     *  2    .THIS IS DONE BY STARTING AT NODE I AND FOLLOWING ITS PREDECESSOR POINT-  *  1578
     *  3    .ER, THEN THE PREDECESSOR'S PREDECESSOR POINTER, ETC., UNTIL NODE K IS    *  1579
     *  4    .REACHED. EACH TIME A NEW PREDECESSOR IS REACHED, A NEW SOLUTION←VECTOR   *  1580
     *  5    .IS CREATED. ITS APPROPRIATE VALUES ARE ENTERED, AND IT IS LINKED TO      *  1581
     *  6    .ANY OTHER SOLUTION←VECTORS THAT REFER TO THE SAME NODE. AT THE SAME      *  1582
     *  7    .TIME AS THE SOLUTION←VECTORS ARE BEING CONSTRUCTED, THE PATH'S ROUTE←   *  1583
     *  8    .VECTOR IS ALSO BEING MADE. THIS IS BECAUSE MANY OF ITS ENTRIES ARE      *  1584
     *  9    .ACCUMULATIONS OF INFORMATION GAINED AT EACH NODE.                       *  1585
     *  10   . ONE WORD OF NOTE, FOR UNIFORMITY, THE SOLUTION←VECTORS OF ALL FOR-     *  1586
     *  11   .WARD PATHS REMAIN AS THEY ARE CONSTRUCTED, BUT THE BACKWARD PATH SOL-   *  1587
     *  12   .UTION←VECTORS HAVE THEIR ORDER REVERSED, THUS, THERE IS NO DATA BASE    *  1588
     *  13   .DISTINCTION BETWEEN BACKWARD AND FORWARD PATHS.                         *  1589
     *  14   .                                                                        *  1590
     *  15   INCREMENT ROUTE NUMBER COUNTER                                          *  1591
     *  16   SET THE UNIT NUMBER                                                     *  1592
     *  17   SET TOTAL ROUTE TIME TO 0                                               *  1593
     *  18   SET TOTAL RISK MEASURE TO 0                                             *  1594
     *  19   SET NUMBER OF NODES TO 1                                                *  1595
     *  20   SET ROUTE MOVEMENT TYPE                                                 *  1596
     *  21   SET THE STARTING TIME                                                   *  1597
     *  22   SET J TO I                                                              *  1598
     *  23   do UNTIL NODE K IS REACHED                                              *  1599
     *  24       INCREMENT SOLUTION←VECTOR COUNTER                                   *  1600
     *  25       SET NODE NUMBER OF SOLUTION←VECTOR TO J                             *  1601
     *  26       SET LANE NUMBER AND PARK TIME                                       *  1602
     *  27       SET CODE FOR CONFLICT RESOLUTION CONSIDERATION                      *  1603
     *  28       INCREMENT ROUTE DISTANCE BY LINK DISTANCE FROM J TO PREDECESSOR NODE *  1604
     *  29       INCREMENT TOTAL ROUTE TIME BY LINK TIME VALUE                       *  1605
     *  30       INCREMENT TOTAL RISK BY LINK RISK VALUE                             *  1606
     *  31       SET TIME IN ACCORDING TO ROUTE TYPE (FORWARD OR BACKWARD)           *  1607
     *  32       SET TIME OUT ACCORDING TO ROUTE TYPE                                *  1608
```

```
* 33          INCREMENT NUMBER OF SOLUTION←VECTORS                          *  1609
* 34          SET J TO ITS PREDECESSOR                                      *  1610
* 35          if J IS NOW K                                                 *  1611
* 36              undo UNTIL NODE K IS REACHED                              *  1612
* 37          endif J IS NOW K                                              *  1613
* 38      enddo UNTIL NODE K IS REACHED                                     *  1614
* 39      .                                                                 *  1615
* 40      .REVERSE THE ORDER OF THE SOLUTION←VECTORS IF NECESSARY           *  1616
* 41      .                                                                 *  1617
* 42      if MOVEMENT TYPE IS BACKWARD                                      *  1618
* 43          REVERSE ORDER OF SOLUTION←VECTORS                             *  1619
* 44      endif MOVEMENT TYPE IS BACKWARD                                   *  1620
* 45      .                                                                 *  1621
* 46      if SOLUTION←VECTOR POINTER OF NODE←VECTOR = 0                     *  1622
* 47          SET SOLUTION←VECTOR POINTER TO SOLUTION←VECTOR COUNTER        *  1623
* 48          SET MULTIPLE USE OF SOLUTION←VECTOR TO -1                     *  1624
* 49      else POINTER DOESN'T EQUAL 0                                      *  1625
* 50          do FOREVER                                                    *  1626
* 51              FOLLOW POINTER                                            *  1627
* 52              if MULTIPLE USE OF POINTER = -1                           *  1628
* 53                  SET MULTIPLE USE OF POINTER TO SOLUTION←VECTOR COUNTER *  1629
* 54                  SET MULTIPLE USE OF SOLUTION←VECTOR TO -1              *  1630
* 55                  undo FOREVER                                          *  1631
* 56              endif MULTIPLE USE OF POINTER = -1                        *  1632
* 57          enddo FOREVER                                                 *  1633
* 58      endif SOLUTION←VECTOR POINTER OF NODE←VECTOR = 0                  *  1634
* 59      SET ROUTE LIST HEAD POINTER                                       *  1635
* 60      if MOVEMENT TYPE IS FORWRD                                        *  1636
* 61          ADD PARKING TIMES WHERE APPROPRIATE                           *  1637
* 62      endif MOVEMENT TYPE IS FORWARD                                    *  1638
* 63      PROMPT FOR ENTRY OF PRINTING OPTION                               *  1639
* 64      if USER WANTS PRINTING                                            *  1640
* 65          PRINT CONTENTS OF SOLUTION←VECTORS                            *  1641
* 66          PRINT ANY TIME CONFLICTS                                      *  1642
* 67      endif USER WANTS PRINTING                                         *  1643
* 68      return TO CALLING PROGRAM                                         *  1644
*                                                                           *
*****************************************************************************
```

CALCULATE SECOND BEST PATH

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . THIS IS DONE BY SYSTEMATICALLY 'ERASING' NODES ALONG THE BEST PATH   *     1646
     *  2    .AND CALCULATING THE NEW BEST PATH. AFTER THE NEW PATH IS CALCULATED,   *     1647
     *  3    .THE ERASED NODE IS RESTORED, AND THE NEXT ERASED, AND SO ON. THE NEW   *     1648
     *  4    .PATH WITH THE COMPLETION TIME CLOSEST TO THE INITIAL BEST PATH IS THEN *     1649
     *  5    .THE SECOND BEST PATH. FOR ADDED INSIGHT, WE WILL KEEP THE NEXT THREE   *     1650
     *  6    .BEST PATHS, HOWEVER, IT IS OFTEN THE CASE THAT TWO (OR EVEN ALL THREE) *     1651
     *  7    .OF THE NEXT BEST PATHS ARE IDENTICAL.                                  *     1652
     *  8    .                                                                       *     1653
     *  9    do UPON FUNCTION REQUEST                                                *     1654
  54 * 10        PURGE ALL ROUTES                                                    *     1655
     * 11        .                                                                   *     1656
     * 12        .THIS ROUTINE CREATES THREE FICTITIOUS UNITS TO BE ASSOCIATED WITH THE *  1657
     * 13        .THREE NEW CALCULATED ROUTES. CHECK FOR STORAGE ROOM IN THE UNITS DATA *  1658
     * 14        .BASE.                                                              *     1659
     * 15        .                                                                   *     1660
     * 16        if THERE IS NOT ENOUGH ROOM IN UNITS FOR THREE MORE UNIT←VECTORS    *     1661
     * 17            PRINT ERROR MESSAGE                                             *     1662
     * 18            return TO CALLING PROGRAM                                       *     1663
     * 19        endif THERE IS NOT ENOUGH ROOM IN UNITS FOR THREE MORE UNIT←VECTORS *     1664
     * 20        PROMPT FOR UNIT NAME                                                *     1665
     * 21        if INPUT IS NULL                                                    *     1666
     * 22            return TO CALLING PROGRAM                                       *     1667
     * 23        endif INPUT IS NULL                                                 *     1668
     * 24        if ENTERED UNIT NAME IS NOT IN THE DATA BASE                        *     1669
     * 25            PRINT ERROR MESSAGE                                             *     1670
     * 26            TRY AGAIN                                                       *     1671
     * 27        else ENTERED UNIT NAME IS LISTED                                    *     1672
     * 28            PROMPT FOR STARTING TIME                                        *     1673
     * 29            PROMPT FOR RISK AND TIME PARAMETERS ...C AND K                  *     1674
     * 30            if DESTINATION IS FEBA AND NONE HAS BEEN DEFINED                *     1675
     * 31                PRINT ERROR MESSAGE                                         *     1676
     * 32                return TO CALLING PROGRAM                                   *     1677
```

```
    * 33          endif DESTINATION IS FEBA                                          *    1678
    * 34          CALCULATE BEST PATH ...FORWARD MOVEMENT WITH NO DECONFLICTING      *    1679
 59 * 35          COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR                          *    1680
    * 36          .                                                                  *    1681
    * 37          .NOW SYSTEMATICALLY ERASE NODES AND CALCULATE NEW PATHS            *    1682
    * 38          .                                                                  *    1683
    * 39          INITIALIZE ARRAY TO CONTAIN THREE NEAREST TIMES TO 32767           *    1684
    * 40          do FOR SECOND THROUGH NEXT-TO-LAST NODES OF THE BEST PATH          *    1685
    * 41              ERASE ROADS LEADING TO THE NODE                                *    1686
 55 * 42              REINITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES                 *    1687
    * 43              CALCULATE BEST PATH ...FORWARD WITH NO DECONFLICTING           *    1688
    * 44              COMPARE COMPLETION TIME WITH OTHERS IN TIME ARRAY               *    1689
    * 45              if THIS TIME IS LESS THAN ANY ONE TIME IN ARRAY                *    1690
    * 46                  PLACE THIS TIME IN ARRAY                                   *    1691
    * 47                  ORDER ARRAY IN ASCENDING FASHION                           *    1692
    * 48                  REMEMBER ERASED NODE                                       *    1693
    * 49              endif THIS TIME IS LESS THAN ANY ONE TIME IN ARRAY             *    1694
    * 50              RESTORE ERASED NODE'S ROADS                                    *    1695
    * 51          enddo FOR SECOND THROUGH NEXT-TO-LAST NODES OF THE BEST PATH       *    1696
    * 52          .                                                                  *    1697
    * 53          .CALCULATE THE THREE NEXT BEST PATHS AND ROUTES                    *    1698
    * 54          .                                                                  *    1699
    * 55          do FOR EACH ENTRY IN TIME ARRAY                                    *    1700
    * 56              ERASE REMEMBERED NODE                                          *    1701
 55 * 57              REINITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES                 *    1702
    * 58              CALCULATE BEST PATH ...FORWARD WITH NO DECONFLICTING           *    1703
 59 * 59              COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR                      *    1704
    * 60              CREATE NEW UNIT←VECTOR                                         *    1705
    * 61              ASSOCIATE NEW UNIT←VECTOR WITH THIS ROUTE                      *    1706
    * 62              PRINT MAP←NUMBER, NODE←NUMBER OF ERASED NODE                   *    1707
    * 63              PRINT UTM LATITUDE, LONGITUDE OF ERASED NODE                   *    1708
    * 64              PRINT DELAY TIME                                               *    1709
    * 65              RESTORE ERASED NODE                                            *    1710
    * 66          enddo FOR EACH ENTRY IN TIME ARRAY                                 *    1711
    * 67      endif ENTERED UNIT NAME IS NOT IN THE DATA BASE                        *    1712
    * 68  enddo UPON FUNCTION REQUEST                                               *    1713
    *                                                                               *
*****************************************************************************************
```

FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . THIS PROBLEM IS: GIVEN N UNITS TRAVELING THROUGH A ROAD NETWORK, AND  *     1715
     *  2    .GIVEN THAT WHEN ANY TWO UNITS WISH TO USE THE SAME NODE DURING OVER-   *     1716
     *  3    .LAPPING TIME INTERVALS THE ONE WITH THE HIGHER PRIORITY USES IT FIRST   *     1717
     *  4    .WHILE THE OTHER WAITS OR GOES AROUND, WHAT ORDERING OF PRIORITIES PRO- *     1718
     *  5    .DUCES THE BEST RESULTS?                                                *     1719
     *  6    . THIS ROUTINE ANSWERS THIS QUESTION BY TRYING EVERY POSSIBLE COMBINA-  *     1720
     *  7    .TION OF PRIORITY ORDERINGS, AND SEEING WHICH ONE DOES IN FACT PRODUCE  *     1721
     *  8    .THE BEST RESULTS. BY 'BEST' WE MEAN THAT THE EARLIEST START TIME OF    *     1722
     *  9    .THE UNITS IN THE CHOSEN PRIORITY SCHEME IS LATER THAN ALL OTHER EAR-   *     1723
     * 10    .LIEST TIMES PRODUCED USING THE OTHER PRIORITY SCHEMES.                 *     1724
     * 11    .                                                                       *     1725
     * 12    do UPON FUNCTION REQUEST                                                *     1726
  54 * 13        PURGE ALL ROUTES                                                    *     1727
     * 14        PROMPT FOR TIME MOVEMENT ...FORWARD GIVES RATHER UNINTERESTING RESULTS *  1728
     * 15        if MOVEMENT IS FORWARD                                              *     1729
     * 16            PROMPT FOR STARTING TIME                                        *     1730
     * 17        else MOVEMENT IS BACKWARD                                           *     1731
     * 18            if DESTINATION IS THE FEBA AND A FEBA HAS NOT BEEN DEFINED      *     1732
     * 19                PRINT ERROR MESSAGE                                         *     1733
     * 20                TRY AGAIN                                                   *     1734
     * 21            endif DESTINATIN IS THE FEBA                                    *     1735
     * 22            PROMPT FOR ARRIVAL TIME                                         *     1736
     * 23        endif MOVEMENT IS FORWARD                                           *     1737
     * 24        .                                                                   *     1738
     * 25        .OBTAIN UNIT NAMES. AT THIS TIME, THIS ROUTINE WILL HANDLE AT MOST FIVE *  1739
     * 26        .UNITS. THIS IS BECAUSE OF TIME CONSIDERATIONS.                     *     1740
     * 27        .                                                                   *     1741
     * 28        do AT MOST FIVE TIMES                                               *     1742
     * 29            PROMPT FOR UNIT NAME                                            *     1743
     * 30            if INPUT IS NULL AND NO UNIT NAME HAS BEEN ENETERED             *     1744
     * 31                return TO CALLING PROGRAM                                   *     1745
     * 32            endif INPUT IS NULL AND NO UNIT NAME HAS BEEN ENTERED           *     1746
```

```
    * 33          if INPUT IS NULL AND AT LEAST ONE UNIT NAME HAS BEEN ENTERED                    *  1747
    * 34              undo AT MOST FIVE TIMES                                                     *  1748
    * 35          else INPUT IS NOT NULL                                                          *  1749
    * 36              if ENTERED UNIT NAME IS NOT IN THE DATA BASE                                *  1750
    * 37                  PRINT MESSAGE                                                           *  1751
    * 38                  TRY AGAIN                                                               *  1752
    * 39              endif ENTERED UNIT NAME IS NOT IN THE DATA BASE                             *  1753
    * 40              REMEMBER UNIT NAME                                                          *  1754
    * 41          endif INPUT IS NULL AND AT LEAST ONE UNIT NAME HAS BEEN ENTERED                 *  1755
    * 42      enddo AT MOST FIVE TIMES                                                            *  1756
    * 43      .                                                                                   *  1757
    * 44      .GENERATE ALL POSSIBLE PRIORITY COMBINATIONS AND MAKE PATH CALCULATIONS             *  1758
    * 45      .USING TIME.                                                                        *  1759
    * 46      .                                                                                   *  1760
    * 47      INITIALIZE EARLIEST TIME                                                            *  1761
    * 48      do FOR EACH COMBINATION OF PRIORITIES                                               *  1762
    * 49          RESET NODE←VECTOR SOLUTION←VECTOR POINTERS                                      *  1763
    * 50          RESET THIS COMBINATION'S EARLIEST START TIME                                    *  1764
    * 51          do FOR EACH UNIT                                                                *  1765
    * 52              CALCULATE BEST PATH ...WITH DECONFLICTING                                   *  1766
    * 53              MAKE SOLUTION←VECTORS AND ROUTE←VECTOR                                      *  1767
    * 54              if THE START TIME IS EARLIER THAN THE OTHERS                                *  1768
    * 55                  REMEMBER THE TIME                                                       *  1769
    * 56              endif THIS PATH WORSE THAN OTHERS                                           *  1770
    * 57          enddo FOR EACH UNIT                                                             *  1771
    * 58          if EARLIEST START TIME IS LATER THAN ALL OTHER EARLIEST TIMES                   *  1772
    * 59              REMEMBER PRIORITY COMBINATION                                               *  1773
    * 60              REMEMBER EARLIEST START TIME                                                *  1774
    * 61          endif EARLIEST START TIME IS LATER THAN ALL OTHER EARLIEST TIMES                 *  1775
    * 62      enddo FOR EACH COMBINATION OF PRIORITIES                                            *  1776
    * 63      .                                                                                   *  1777
    * 64      .CALCULATE RESULTS                                                                  *  1778
    * 65      .                                                                                   *  1779
    * 66      REINITIALIZE SOLUTION←VECTORS AND ROUTE←VECTORS                                     *  1780
    * 67      do FOR EACH UNIT USING THE BEST PRIORITY SCHEME                                     *  1781
    * 68          CALCULATE BEST PATH                                                             *  1782
 59 * 69          COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR                                       *  1783
    * 70      enddo FOR EACH UNIT USING THE BEST PRIORITY SCHEME                                  *  1784
```

```
* 71        PRINT MESSAGE INFORMING USER THAT RESULTS ARE AVAILABLE                    *    1785
* 72    enddo UPON FUNCTION REQUEST                                                    *    1786
*                                                                                      *
************************************************************************************
```

CALCULATE BEST NODE AT WHICH TO INTERDICT

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . THIS ROUTINE CALCULATES THE NODE AT WHICH IT IS BEST TO INTERDICT,    *  1788
     *  2    .WHERE 'BEST' MEANS 'HAS THE GREATEST WORTH DIFFERENCE BETWEEN THE BEST *  1789
     *  3    .PATH AND THE PATH WITH INTERDICTION'. IN ADDITION, THE PROGRAM FINDS   *  1790
     *  4    .THE TIME WINDOW DURING WHICH THE INTERDICTION SHOULD OCCUR.            *  1791
     *  5    . THIS IS DONE IN A MANNER SIMILAR TO THE PROCESS FOR CALCULATING THE   *  1792
     *  6    .SECOND BEST PATH. EACH NODE OF THE BEST PATH IS SYSTEMATICALLY ER-     *  1793
     *  7    .ASED, BUT THEN THERE IS A DIFFERENCE. THIS ROUTINE FORCES THE UNIT TO  *  1794
     *  8    .TRAVEL A DISTANCE DOWN THE BEST PATH, AND THEN LETS IT GO ITS OWN WAY. *  1795
     *  9    .E.G., SUPPOSE THAT THE FIFTH NODE HAS BEEN ERASED. THERE WILL BE FOUR  *  1796
     *  10   .PATHS CALCULATED FOR THIS ONE NODE ERASURE. FIRST, THE UNIT WILL BE    *  1797
     *  11   .FORCED TO TRAVEL THROUGH NODE ONE, AND THEN IT MAY GO AS IT PLEASES.   *  1798
     *  12   .NEXT, IT IS FORCED THROUGH NODE 2, THEN THROUGH NODE 3, AND FINALLY,   *  1799
     *  13   .THROUGH NODE FOUR. IN THIS WAY, WE SIMULATE TIME RELATED INTERDICTION. *  1800
     *  14   .                                                                       *  1801
     *  15   do UPON FUNCTION REQUEST                                                *  1802
  54 *  16       PURGE ALL ROUTES                                                    *  1803
     *  17       .                                                                   *  1804
     *  18       .WE WILL ACTUALLY KEEP THE THREE BEST INTERDICTION CASES IN AN ARRAY. *  1805
     *  19       .AND, WE WILL CREATE THREE FICTITIOUS UNITS.                        *  1806
     *  20       .                                                                   *  1807
     *  21       INITIALIZE BEST ARRAY                                               *  1808
     *  22       if THERE IS NOT ROOM FOR THREE MORE UNIT←VECTORS                     *  1809
     *  23           PRINT MESSAGE                                                   *  1810
     *  24           return TO CALLING PROGRAM                                       *  1811
     *  25       endif THERE IS NOT ROOM FOR THREE MOR UNIT←VECTORS                   *  1812
     *  26       PROMPT FOR UNIT NAME                                                *  1813
     *  27       if INPUT IS NULL                                                    *  1814
     *  28           return TO CALLING PROGRAM                                       *  1815
     *  29       else INPUT IS NOT NULL                                              *  1816
     *  30           if UNIT NAME IS NOT IN DATA BASE                                *  1817
     *  31               PRINT MESSAGE                                               *  1818
     *  32               TRY AGAIN                                                   *  1819
```

```
      * 33          endif UNIT NAME IS NOT IN DATA BASE                          *    1820
      * 34        endif INPUT IS NULL                                            *    1821
      * 35        .                                                              *    1822
      * 36        .GET PARAMETERS                                                *    1823
      * 37        .                                                              *    1824
      * 38        PROMPT FOR START TIME                                          *    1825
      * 39        PROMPT FOR RISK AND TIME PARAMETERS ...C AND K                 *    1826
      * 40        CALCULATE BEST PATH                                            *    1827
   59 * 41        COMPUTE SOLUTION+VECTORS AND ROUTE+VECTOR                      *    1828
      * 42        .                                                              *    1829
      * 43        .SYSTEMATICALLY ERASE NODES, AND FORCE UNIT TO TRAVEL          *    1830
      * 44        .                                                              *    1831
      * 45        do FOR THE SECOND THROUGH NEXT-TO-LAST NODES OF THE BEST PATH  *    1832
      * 46            ERASE NODE                                                 *    1833
      * 47            do FOR EACH NODE FROM FIRST NODE TO PREDECESSOR OF ERASED NODE *    1834
      * 48                LOOK IN SOLUTION+VECTOR OF NODE TO OBTAIN START TIME   *    1835
      * 49                .                                                      *    1836
      * 50                .WE'LL ONLY CALCULATE FROM THIS NODE TO THE DESTINATION *    1837
      * 51                .                                                      *    1838
      * 52                CALCULATE BEST PATH OF THIS NODE TO THE DESTINATION    *    1839
      * 53                if PATH'S WORTH IS GREATER THAN ANY OF THOSE IN THE ARRAY *    1840
      * 54                    PLACE THIS WORTH IN WORTH ARRAY                    *    1841
      * 55                    REMEMBER INTERDICTED NODE                          *    1842
      * 56                    REMEMBER 'FORCED-TO' NODE                          *    1843
      * 57                endif PATH'S WORTH IS GREATER THAN ANY OF THOSE IN THE ARRAY *    1844
      * 58            enddo FOR EACH NODE FROM FIRST NODE TO PREDECESSOR OF ERASED NODE *    1845
      * 59            RESTORE ERASED NODE                                        *    1846
      * 60        enddo FOR THE SECOND THROUGH NEXT-TO-LAST NODE OF BEST PATH    *    1847
      * 61        .                                                              *    1848
      * 62        .CALCULATE CHOSEN PATHS. EACH PATH WILL BE THE COMBINATION OF TWO *    1849
      * 63        .PATHS, ONE FROM THE FIRST NODE TO THE INTERDICTED NODE, THEN ANOTHER *    1850
      * 64        .FROM THE INTERDICTED NODE TO THE DESTINATION.                 *    1851
      * 65        .                                                              *    1852
      * 66        do FOR EACH ENTRY IN WORTH ARRAY                               *    1853
      * 67            ERASE REMEMBERED NODE                                      *    1854
      * 68            INITIALIZE 'WORKING LIST' NODE+VECTOR ENTRIES              *    1855
      * 69            CALCULATE BEST PATH FROM START NODE TO PREDECESSOR OF ERASED NODE *    1856
   59 * 70            COMPUTE SOLUTION+VECTORS AND ROUTE+VECTOR                  *    1857
```

```
     * 71            INITIALIZE 'WORKING LIST' NODE<VECTOR ENTRIES                                    *    1858
     * 72            CALCULATE BEST PATH FROM PREDECESSOR TO DESTINATION NODE                         *    1859
  59 * 73            COMPUTE SOLUTION<VECTORS AND ROUTE<VECTOR                                        *    1860
     * 74            COMBINE SOLUTION<VECTORS AND ROUTE<VECTORS TO MAKE ONE ROUTE                     *    1861
     * 75            CREATE NEW UNIT ASSOCIATED WITH THIS ROUTE                                       *    1862
     * 76            PRINT MAP<NUMBER,NODE<NUMBER OF INTERDICTED NODE                                 *    1863
     * 77            PRINT MAP<NUMBER, NODE<NUMBER OF 'FORCED' NODE                                   *    1864
     * 78            PRINT TIME WINDOW                                                                *    1865
     * 79            PRINT TIME DELAY                                                                 *    1866
     * 80          enddo FOR EACH ENTRY IN WORTH ARRAY                                                *    1867
     * 81          return TO CALLING PROGRAM                                                          *    1868
     * 82        enddo UPON FUNCTION REQUEST                                                          *    1869
     *                                                                                                *
     **************************************************************************************************
****** 100 MANY LINES IN SEGMENT
```

PRESENT RESULTS

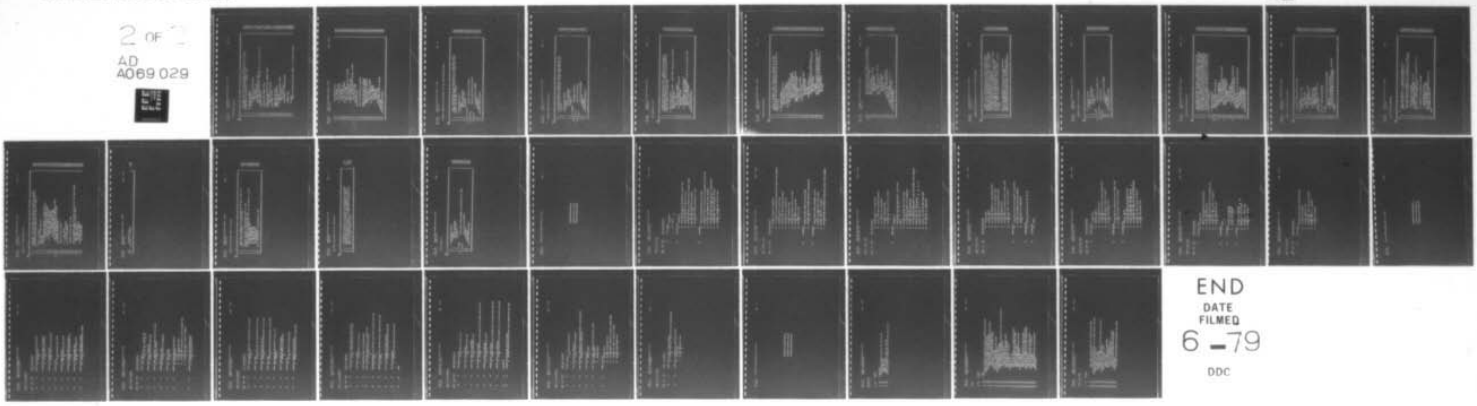
```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . THESE ROUTINES ENABLE THE USER TO SEE THE RESULTS OF THE PATH ALGO-  *  1871
     *  2    .RITHMS AS PRINTED OUTPUT.                                             *  1872
     *  3    .                                                                      *  1873
     *  4    do UPON FUNCTION REQUEST                                               *  1874
     *  5        do FOREVER                                                         *  1875
     *  6            DISPLAY FUNCTION MENU                                          *  1876
     *  7            WAIT FOR USER INPUT                                            *  1877
     *  8            do CASE OF                                                     *  1878
     *  9 TABLE:                                                                    *  1879
  54 * 10                PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES   *  1880
     * 11 HARDCOPY:                                                                *  1881
  65 * 12                PRINT PATH STATISTICS                                     *  1882
     * 13 EXIT:                                                                    *  1883
     * 14                return TO CALLING PROGRAM                                 *  1884
     * 15            enddo CASE OF                                                 *  1885
     * 16        enddo FOREVER                                                     *  1886
     * 17    enddo UPON FUNCTION REQUEST                                           *  1887
     *                                                                              *
     ********************************************************************************
```

PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES

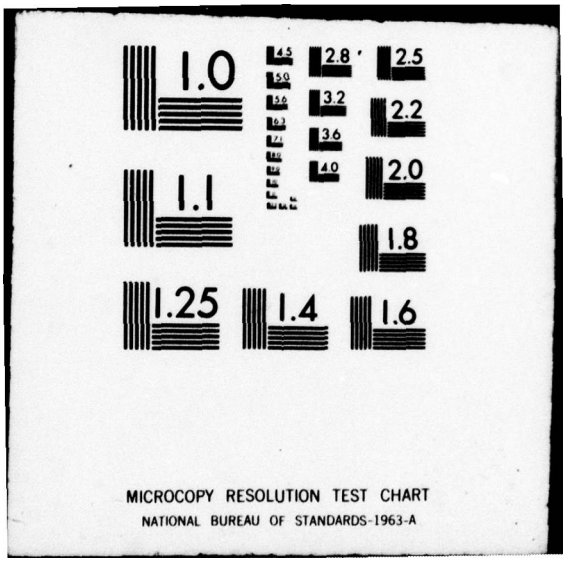
```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . IF THIS FUNCTION IS REQUESTED, A TABLE SHOWING ALL ROUTE NUMBERS, EACH *    1889
     *  2    .WITH THE UNIT NAME WITH WHICH IT IS ASSOCIATED, WILL BE PRINTED.       *    1890
     *  3    .                                                                       *    1891
     *  4    do UPON FUNCTION REQUEST                                                *    1892
     *  5        PRINT TABLE OF ROUTE NUMBERS AND ASSOCIATED UNIT NAMES              *    1893
     *  6        return TO CALLING PROGRAM                                           *    1894
     *  7    enddo UPON FUNCTION REQUEST                                             *    1895
     *                                                                              *
     ********************************************************************************
```

AD-A069 029 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CALIF F/G 9/2
LAMAS SYSTEM MANUAL.(U)
FEB 78 DAAG39-77-C-0112
UNCLASSIFIED NL

2 OF 2
AD
A069 029



END
DATE FILMED
6-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

PRINT PATH STATISTICS

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . THIS ENABLE THE USER TO HAVE HARD-COPY EVIDENCE OF THE CALCULATIONS  *     1897
     *  2    .PERFORMED BY THIS SYSTEM. THIS INCLUDES TOTAL PATH STATISTICS, AS WELL *     1898
     *  3    .AS THE ABILITY TO HAVE OUTPUT IN A SNAPSHOT MODE.                      *     1899
     *  4    .                                                                       *     1900
     *  5    do UPON FUNCTION REQUEST                                                *     1901
     *  6        do FOR EACH UNIT                                                    *     1902
     *  7            if THERE IS NO ASSOCIATED ROUTE                                 *     1903
     *  8                PRINT MESSAGE                                               *     1904
     *  9                cycle FOR EACH UNIT                                         *     1905
     * 10            endif THERE IS NO ASSOCIATED ROUTE                              *     1906
     * 11            PRINT TOTAL PATH STATISTICS ...DISTANCE, TIME, START AND FINISH, ETC.  *     1907
     * 12        enddo FOR EACH UNIT                                                 *     1908
     * 13        .                                                                   *     1909
     * 14        .NOW SEE ABOUT SNAPSHOT MODE                                        *     1910
     * 15        .                                                                   *     1911
     * 16        PROMPT FOR USER DESIRE TO HAVE SNAPSHOT OUTPUT                      *     1912
     * 17        if INPUT IS NULL                                                    *     1913
     * 18            return TO CALLING PROGRAM                                       *     1914
     * 19        endif INPUT IS NULL                                                 *     1915
     * 20        .                                                                   *     1916
     * 21        PROMPT FOR START TIME                                               *     1917
     * 22        PROMPT FOR TIME INCREMENT                                           *     1918
     * 23        do FOR EACH UNIT                                                    *     1919
     * 24            if THERE IS NO ASSOCIATED ROUTE                                 *     1920
     * 25                PRINT MESSAGE                                               *     1921
     * 26                cycle FOR EACH UNIT                                         *     1922
     * 27            endif THERE IS NO ASSOCIATED ROUTE                              *     1923
     * 28            .                                                               *     1924
     * 29            .SNAPSHOT MODE                                                  *     1925
     * 30            .                                                               *     1926
     * 31            do FOREVER                                                      *     1927
     * 32                FIND UNIT POSITION FOR TIME OF INTEREST ...AS IN SNAPSHOT DISPLAY  *     1928
```

```
* 33              if UNIT HAS NOT STARTED                                          *  1929
* 34                  PRINT MESSAGE                                                *  1930
* 35                  cycle FOR EACH UNIT                                          *  1931
* 36              elseif UNIT HAS FINISHED                                         *  1932
* 37                  PRINT MESSAGE                                                *  1933
* 38                  cycle FOR EACH UNIT                                          *  1934
* 39              endif UNIT HAS NOT STARTED                                       *  1935
* 40              if UNIT IS AT A NODE                                             *  1936
* 41                  PRINT STATISTICS FOR UNIT AT THE NODE                        *  1937
* 42                  cycle FOR EACH UNIT                                          *  1938
* 43              else UNIT IS BETWEEN TWO NODES                                   *  1939
* 44                  PRINT STATISTICS FOR UNIT BETWEEN THE NODES                  *  1940
* 45                  cycle FOR EACH UNIT                                          *  1941
* 46              endif UNIT IS AT A NODE                                          *  1942
* 47          enddo FOR EACH UNIT                                                  *  1943
* 48          .                                                                    *  1944
* 49          .SEE IF THE USER WISHES TO CONTINUE                                  *  1945
* 50          .                                                                    *  1946
* 51          PROMPT FOR USER DESIRE TO CONTINUE                                   *  1947
* 52          if USER DOES NOT WANT TO CONTINUE                                    *  1948
* 53              PROMPT FOR NEW TIME INCREMENT                                    *  1949
* 54              if INPUT IS NULL                                                 *  1950
* 55                  return TO CALLING PROGRAM                                    *  1951
* 56              else INPUT IS NOT NULL                                           *  1952
* 57                  SET TIME INCREMENT                                           *  1953
* 58                  INCREMENT TIME                                               *  1954
* 59                  cycle FOREVER                                                *  1955
* 60              endif INPUT IS NULL                                              *  1956
* 61          endif USER DOES NOT WANT TO CONTINUE                                 *  1957
* 62      enddo FOREVER                                                            *  1958
* 63  enddo UPON FUNCTION REQUEST                                                  *  1959
*                                                                                  *
************************************************************************************
```

IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE CROSS-COUNTRY DATA BASE

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . THIS IS THE OTHER HALF OF THE LAMAS SYSTEM. IT IS MUCH MORE LIMITED  *     1961
     *  2    .IN SCOPE THAN THE ROAD NETWORK SECTION, AND THIS IS MAINLY DUE TO THE *     1962
     *  3    .NATURE OF THE DATA BASE INVOLVED.                                     *     1963
     *  4    .                                                                      *     1964
     *  5    do UPON FUNCTION REQUEST                                               *     1965
     *  6        do FOREVER                                                         *     1966
     *  7            DISPLAY FUNCTION MENU                                          *     1967
     *  8            WAIT FOR USER INPUT                                            *     1968
     *  9            do CASE OF                                                     *     1969
     * 10 INITIALIZE:                                                              *     1970
  67 * 11                PERFORM TERRAIN INITIALIZATION                           *     1971
     * 12 PATHS:                                                                   *     1972
  71 * 13                PERFORM TERRAIN PATH CALCULATIONS                        *     1973
     * 14 EXIT:                                                                    *     1974
     * 15                return TO CALLING PROGRAM                                *     1975
     * 16            enddo CASE OF                                                 *     1976
     * 17        enddo FOREVER                                                     *     1977
     * 18    enddo UPON FUNCTION REQUEST                                           *     1978
     *                                                                              *
     ********************************************************************************
```

```
     PERFORM TERRAIN INITIALIZATION


REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . IN THE CROSS-COUNTRY AND CONCEALMENT SECTION, THERE ARE ONLY TWO     *     1980
     *  2    .THINGS WHICH NEED TO BE INITIALIZED, THE MAP←NUMBER, AND THE UNITS    *     1981
     *  3    .DATA BASE.                                                            *     1982
     *  4                                                                           *     1983
     *  5    do UPON FUNCTION REQUEST                                               *     1984
     *  6        do FOREVER                                                         *     1985
     *  7            DISPLAY FUNCTION MENU                                          *     1986
     *  8            WAIT FOR USER INPUT                                            *     1987
     *  9            do CASE OF                                                     *     1988
     * 10 MAP←NUMBER:                                                              *     1989
  58 * 11                 MAP←NUMBER TO BE USED                                    *     1990
     * 12 UNITS:                                                                   *     1991
  69 * 13                 ESTABLISH UNIT←VECTORS                                   *     1992
     * 14 EXIT:                                                                    *     1993
     * 15                 return TO CALLING PROGRAM                                *     1994
     * 16             enddo CASE OF                                                *     1995
     * 17         enddo FOREVER                                                    *     1996
     * 18    enddo UPON FUNCTION REQUEST                                           *     1997
     *                                                                              *
     ********************************************************************************
```

MAP#NUMBER TO BE USED

```
REF
PAGE ************************************************************************
     *                                                                      *
     *  1    . FOR THE CROSS-COUNTRY AND CONCEALMENT ALGORITHMS, ONLY ONE MAP MAY   *   1999
     *  2    .BE CONSIDERED AT ONE TIME. THIS IS SO BECAUSE OF THE LARGE NUMBER OF  *   2000
     *  3    .NODES WHICH ONE MAP REPRESENTS (9186), AND THE TIME INVOLOVED FOR A   *   2001
     *  4    .PATH CALCULATION USING THAT MANY NODES.                               *   2002
     *  5    .                                                                      *   2003
     *  6    do UPON FUNCTION REQUEST                                               *   2004
     *  7        READ 'SCMDIR.DAT' INTO MEMORY ...CROSS-COUNTRY AND CONCEALMENT DIRECTORY  *   2005
     *  8        PROMPT FOR MAP#NUMBER                                              *   2006
     *  9        if INPUT IS NULL                                                   *   2007
     * 10            return TO CALLING PROGRAM                                     *   2008
     * 11        elseif INPUT NUMBER IS NOT IN DIRECTORY                            *   2009
     * 12            PRINT ERROR MESSAGE                                           *   2010
     * 13            TRY AGAIN                                                     *   2011
     * 14        else INPUT NUMBER IS IN THE DIRECTORY                              *   2012
     * 15            READ THE MAP'S NODES INTO MEMORY                              *   2013
     * 16            ESTABLISH THE SCREEN EXTREMES ...VALUES ARE IN THE DIRECTORY ENTRY  *   2014
     * 17            return TO CALLING PROGRAM                                     *   2015
     * 18        endif INPUT IS NULL                                                *   2016
     * 19    enddo UPON FUNCTION REQUEST                                            *   2017
     *                                                                      *
     ************************************************************************
```

ESTABLISH UNIT←VECTORS

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . JUST AS IN THE ROAD NETWORK SECTION, THIS DATA BASE MUST BE CON-     *  2019
     *  2    .STRUCTED BY THE USER. THE ONLY DIFFERENCE BETWEEN THE TWO IS THAT     *  2020
     *  3    .HERE, THE START AND DESTINATION ARE DESCRIBED AS UTM LATITUDE AND LON-*  2021
     *  4    .GITUDE PAIRS.                                                         *  2022
     *  5    .                                                                      *  2023
     *  6    do UPON FUNCTION REQUEST                                               *  2024
     *  7        if NO MAP HAS BEEN INITIALIZED                                     *  2025
     *  8            PRINT ERROR MESSAGE                                            *  2026
     *  9            return TO CALLING PROGRAM                                      *  2027
     * 10        else A MAP HAS BEEN INITIALIZED                                    *  2028
     * 11            do UNTIL UNIT←COUNTER EQUALS 60                                *  2029
     * 12                if USER DESIRES RE-INITIALIZATION                          *  2030
     * 13                    CLEAR ALL EXISTING UNIT←VECTORS                        *  2031
     * 14                    RESET UNIT←COUNTER TO ZERO                             *  2032
     * 15                endif USER DESIRES RE--INITIALIZATION                      *  2033
     * 16                PROMPT FOR ENTRY OF UNIT NAME                              *  2034
     * 17                if INPUT IS NULL                                           *  2035
     * 18                    return TO CALLING PROGRAM                              *  2036
     * 19                else INPUT IS NOT NULL                                     *  2037
     * 20                    if THIS UNIT NAME ALREADY EXISTS                       *  2038
     * 21                        PRINT ERROR MESSAGE                                *  2039
     * 22                        TRY AGAIN                                          *  2040
     * 23                    else UNIT NAME IS NEW                                  *  2041
     * 24                        PROMPT FOR UTM COORDINATES OF STARTING NODE        *  2042
     * 25                        if COORDINATES DO NOT LIE WITHIN THE INITIALIZED MAP*  2043
     * 26                            PRINT ERROR MESSAGE                            *  2044
     * 27                            TRY AGAIN                                      *  2045
     * 28                        endif COORDINATES DO NOT LIE WITHIN THE INITIALIZED MAP*  2046
     * 29                        PROMPT FOR UTM COORDINATE OF DESTINATION NODE      *  2047
     * 30                        if COORDINATES DO NOT LIE WITHIN THE INITIALIZED MAP*  2048
     * 31                            PRINT ERROR MESSAGE                            *  2049
     * 32                            TRY AGAIN                                      *  2050
```

```
    * 33                    endif COORDINATES DO NOT LIE WITHIN THE INITIALIZED MAP                *    2051
 70 * 34                    CHANGE ENTERED COORDINATES TO AN INDEX INTO THE DATA BASE              *    2052
    * 35                    MAKE ASSIGNMENTS TO UNIT+VECTOR                                       *    2053
    * 36                    PROMPT FOR PRIORITY                                                   *    2054
    * 37                    if PRIORITY NOT BETWEEN 0 AND 127                                     *    2055
    * 38                        PRINT ERROR MESSAGE                                               *    2056
    * 39                        TRY AGAIN                                                         *    2057
    * 40                    endif PRIORITY NOT BETWEEN 0 AND 127                                  *    2058
    * 41                    PROMPT FOR TYPE CODE                                                  *    2059
    * 42                    if TYPE CODE NOT ALLOWABLE                                            *    2060
    * 43                        PRINT ERROR MESSAGE                                               *    2061
    * 44                        TRY AGAIN                                                         *    2062
    * 45                    endif TYPE CODE NOT ALLOWABLE                                         *    2063
    * 46                endif THIS UNIT NAME ALREADY EXISTS                                       *    2064
    * 47            endif INPUT IS NULL                                                           *    2065
    * 48        enddo UNTIL UNIT+COUNTER EQUALS SIXTY                                             *    2066
    * 49    endif NO MAP HAS BEEN INITIALIZED                                                     *    2067
    * 50 enddo UPON FUNCTION REQUEST                                                              *    2068
    *                                                                                             *
*****************************************************************************************************
```

CHANGE ENTERED COORDINATES TO AN INDEX INTO THE DATA BASE

```
REF
PAGE ***********************************************************************
     *                                                                     *
     *  1    .IN ORDER TO KEEP THE MEMORY USAGE AT A MINIMUM, AN INDEXING SCHEME   *  2070
     *  2    .HAS BEEN DEVISED. THE DATA BASE HAS BEEN PACKED TOGETHER SO THAT     *  2071
     *  3    .ONE WORD CONTAINS FOUR DATA VALUES, OR EVERY DATA VALUE IS REPRESENTED *  2072
     *  4    .BY FOUR BITS. ONE FOUR-BIT AREA IS CALLED A NIBBLE (HALF OF A BYTE).  *  2073
     *  5    .THUS, THERE ARE 8184 NIBBLES IN THE DATA BASE.                       *  2074
     *  6    . THE POINT IS THEN TO MAP THE UTM COORDINATE PAIR TO ONE SQUARE ON THE *  2075
     *  7    .93 X 89 GRID, AND THEN TRANSLATE THAT VALUE TO A NIBBLE VALUE.       *  2076
     *  8    .THE POSSIBLE RANGE OF THE UTM COORDINATE PAIRS IS 1 TO 506           *  2077
     *  9    .(22 * 23, = SQUARE KILOMETERS ON A MAP), SO WE'RE MAPPING 1 TO 506 TO *  2078
     * 10    .1 TO 8184.                                                           *  2079
     * 11    .                                                                     *  2080
     * 12    CALCULATE THE VALUE BETWEEN 1 AND 506 FOR THE GIVEN COORDINATES       *  2081
     * 13    CALCULATE THE PERCENTAGE OF THIS VALUE OF THE WHOLE                   *  2082
     * 14    MULTIPLY THE PERCENTAGE BY 8183 AND ADD 1 TO OBTAIN THE NIBBLE        *  2083
     * 15    ASSIGN THE VALUE TO EITHER STARTING LOCATION OR DESTINATION LOCATION  *  2084
     *                                                                     *
     ***********************************************************************
```

PERFORM TERRAIN PATH CALCULATIONS

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    do UPON FUNCTION REQUEST                                               *  2086
     *  2        do FOREVER                                                         *  2087
     *  3            DISPLAY FUNCTION MENU                                          *  2088
     *  4            WAIT FOR USER INPUT                                            *  2089
     *  5            do CASE OF                                                     *  2090
     *  6 PATH:                                                                     *  2091
  72 *  7                CALCULATE CROSS-COUNTRY PATH                               *  2092
     *  8 SOLUTION:                                                                 *  2093
  78 *  9                DISPLAY RESULTS                                            *  2094
     * 10 EXIT:                                                                     *  2095
     * 11                return TO THE CALLING PROGRAM                              *  2096
     * 12            enddo CASE OF                                                  *  2097
     * 13        enddo FOREVER                                                      *  2098
     * 14    enddo UPON FUNCTION REQUEST                                            *  2099
     *                                                                              *
     ********************************************************************************
```

CALCULATE CROSS-COUNTRY PATH

```
REF
PAGE ************************************************************************************************
     *                                                                                                *
     *  1    . THIS ROUTINE CALCULATES THE PATH OF A USER-ENTERED UNIT USING THE                       *    2101
     *  2    .CROSS-COUNTRY DATA BASE. THE MAJOR DIFFERENCE BETWEEN THIS ROUTINE AND                  *    2102
     *  3    .THE ROUTINE FOR THE ROAD NETWORK IS THAT THERE IS NOW AN ARRAY CALLED                   *    2103
     *  4    .THE WORKING LIST. PREVIOUSLY, THE WORKING LIST VARIABLES WERE STORED                    *    2104
     *  5    .WITHIN THE NODE*VECTOR, BUT NOW, WITH 8184 NODES TO BE CONSIDERED, IT                   *    2105
     *  6    .IS TOTALLY IMPRACTICAL TO TRY TO RETAIN SPACE FOR EACH NODE'S WORKING                   *    2106
     *  7    .LIST ENTRIES. THUS, A WORKING LIST ARRAY EXISTS. IT IS TREATED AS A                     *    2107
     *  8    .LIST OF VECTORS, BEING USED IN ORDER (FROM INDEX 1 TO N) AS THEY ARE                    *    2108
     *  9    .NEEDED BY THE ALGORITHM.                                                                *    2109
     *  10   .                                                                                        *    2110
     *  11   do UPON FUNCTION REQUEST                                                                 *    2111
  54 *  12       PURGE ALL ROUTES                                                                     *    2112
  73 *  13       OBTAIN UNIT NAMES TO BE CONSIDERED                                                   *    2113
     *  14       do FOR EACH UNIT                                                                     *    2114
  74 *  15           OBTAIN THE PATH PARAMETERS                                                       *    2115
     *  16           if THERE IS AN ERROR RETURN                                                      *    2116
     *  17               GO BACK TO OBTAIN UNIT NAMES AND TRY AGAIN                                   *    2117
     *  18           endif THERE IS AN ERROR RETURN                                                   *    2118
  75 *  19           CALCULATE THE CROSS-COUNTRY PATH                                                 *    2119
     *  20           if THE PATH COULD NOT COMPLETE                                                   *    2120
     *  21               PRINT MESSAGE                                                                *    2121
     *  22               cycle FOR EACH UNIT                                                          *    2122
     *  23           else THE PATH DID COMPLETE                                                       *    2123
  77 *  24               CONSTRUCT SOLUTION*VECTORS AND ROUTE*VECTOR                                  *    2124
     *  25           endif THE PATH COULD NOT COMPLETE                                                *    2125
     *  26       enddo FOR EACH UNIT                                                                  *    2126
     *  27       PROMPT FOR USER WISH FOR SOLUTION*VECTOR PRINTOUT                                    *    2127
     *  28       if USER WANTS OUTPUT                                                                 *    2128
     *  29           PRINT SOLUTION*VECTORS                                                           *    2129
     *  30       endif USER WANTS OUTPUT                                                              *    2130
     *  31       return TO CALLING PROGRAM                                                            *    2131
     *  32   enddo UPON FUNCTION REQUEST                                                             *    2132
     *                                                                                               *
     ************************************************************************************************
```

OBTAIN UNIT NAMES TO BE CONSIDERED

```
REF
PAGE ************************************************************************
     *                                                                      *
     *  1    . THE USER MAY ENTER AT MOST SIXTY NAMES.                      *  2134
     *  2    .                                                              *  2135
     *  3    do SIXTY TIMES                                                 *  2136
     *  4        PROMPT ENTRY OF UNIT NAME                                  *  2137
     *  5        if NULL INPUT                                              *  2138
     *  6            undo SIXTY TIMES                                       *  2139
     *  7        endif NULL INPUT                                           *  2140
     *  8        if INPUT NAME DOES NOT EXIST                               *  2141
     *  9            PRINT ERROR MESSAGE                                    *  2142
     * 10            TRY AGAIN                                              *  2143
     * 11        else UNIT NAME DOES EXIST                                  *  2144
     * 12            INCREMENT UNIT COUNTER                                 *  2145
     * 13            REMEMBER THE UNIT NAME                                 *  2146
     * 14        endif INPUT NAME DOES NOT EXIST                            *  2147
     * 15    enddo SIXTY TIMES                                              *  2148
     * 16    .                                                              *  2149
     * 17    .ALL UNIT NAME HAVE BEEN ENTERED.                              *  2150
     * 18    .                                                              *  2151
     * 19    do ONCE                                                        *  2152
     * 20        if LESS THAN TWO UNIT NAMES HAVE BEEN ENTERED              *  2153
     * 21            undo ONCE                                              *  2154
     * 22        else MORE THAN ONE NAME HAS BEEN ENTERED                   *  2155
     * 23            SORT UNIT NAMES ACCORDING TO THEIR PRIORITY ...BUBBLE SORT  2156
     * 24        endif LESS THAN TWO UNIT NAMES HAVE BEEN ENTERED           *  2157
     * 25    enddo ONCE                                                     *  2158
     *                                                                      *
     ************************************************************************
```

OBTAIN THE PATH PARAMETERS

```
REF
PAGE ***************************************************************************************
     *                                                                                     *
     *  1    do                                                                            *    2160
     *  2        PROMPT FOR MOVEMENT TYPE ...BACKWARDS OR FORWARDS IN TIME                 *    2161
     *  3        PROMPT FOR START TIME ...DEPENDENT UPON MOVEMENT TYPE                     *    2162
     *  4        .                                                                         *    2163
     *  5        .REINITIALIZATION OF THE WORKING LIST IS NEEDED ONLY IF EITHER THE        *    2164
     *  6        .UNIT TYPE, START TIME, MOVEMENT TYPE, OR START LOCATION HAS CHANGED.     *    2165
     *  7        .                                                                         *    2166
     *  8        if REINITIALIZATION IS NEEDED                                             *    2167
     *  9            SET WORKING LIST ARRAY ENTRIES TO ZERO                                *    2168
     * 10        endif REINITIALIZATION IS NEEDED                                          *    2169
     * 11        ESTABLISH THE BEGINNING NODE (=K), AND THE DESTINATION NODE (=I)          *    2170
     * 12        SET J TO K ...J WILL TRACE THE PATH                                       *    2171
     * 13        .                                                                         *    2172
     * 14        .SET CUMMULATIVE TIME AND WORTH FOR NODE J.                               *    2173
     * 15        .                                                                         *    2174
     * 16        if J HAS NO WORKING LIST ENTRY                                            *    2175
     * 17            CREATE NEW WORKING LIST ENTRY                                         *    2176
     * 18        endif J HAS NO WORKING LIST ENTRY                                         *    2177
     * 19        SET CUMMULATIVE TIME TO ENTERED BEGINNING TIME                            *    2178
     * 20        SET CUMMULATIVE WORTH TO 0                                                *    2179
     * 21    enddo                                                                         *    2180
     *                                                                                     *
     ***************************************************************************************
```

CALCULATE THE CROSS-COUNTRY PATH

```
REF
PAGE ***********************************************************************************
     *                                                                                 *
     *  1    . GIVEN NODE J, THE LAST NODE TO BE LABELED, THIS ROUTINE CALCULATES      *   2182
     *  2    .ITS ADJACENT NODES TIME VALUES. THEN THE NEXT NODE TO BE LABELED IS      *   2183
     *  3    .SELECTED AND THE PROCESS CONTINUES UNTIL NODE I IS LABELED.              *   2184
     *  4    .                                                                         *   2185
     *  5    do UNTIL I IS REACHED                                                     *   2186
     *  6        FIND NODE J IN THE WORKING LIST                                       *   2187
     *  7        LABEL NODE J                                                          *   2188
     *  8        do FOR EACH ADJACENT NODE OF J ...EIGHT IN ALL                        *   2189
     *  9            if THE ADJACENT NODE HAS NO WORKING LIST ENTRY                    *   2190
     * 10                CREATE AN ENTRY FOR THE ADJACENT NODE                        *   2191
     * 11            else THE NODE HAS A WORKING LIST ENTRY                           *   2192
     * 12                if THE ADJACENT NODE IS LABELED                              *   2193
     * 13                    cycle FOR EACH ADJACENT NODE                             *   2194
     * 14                endif THE ADJACENT NODE IS LABELED                           *   2195
     * 15            endif THE ADJACENT NODE HAS NO WORKING LIST ENTRY                 *   2196
  75 * 16            CALCULATE THE ADJACENT NODE'S TIME VALUES                        *   2197
     * 17        enddo FOR EACH ADJACENT NODE                                          *   2198
     * 18        .                                                                     *   2199
     * 19        .LOOK FOR I BEING LABELED                                             *   2200
     * 20        .                                                                     *   2201
     * 21        if I IS NOW LABELED                                                   *   2202
     * 22            undo UNTIL I IS REACHED                                           *   2203
     * 23        endif IS IS NOW LABELED                                               *   2204
     * 24        .                                                                     *   2205
     * 25        .CALCULATE THE NEXT NODE TO LABEL                                     *   2206
     * 26        .                                                                     *   2207
     * 27        FIND THE UNLABELED WORKING LIST ENTRY WHICH HAS THE LEAST WORTH MEASURE *   2208
     * 28        if NONE EXISTS ...THE ROUTE COULDN'T FINISH                           *   2209
     * 29            PRINT ERROR MESSAGE                                               *   2210
     * 30            undo UNTIL I IS REACHED                                           *   2211
     * 31        else AN ENTRY WAS FOUND                                               *   2212
     * 32            SET J TO THIS ENTRY                                               *   2213
```

```
* 33        endif NONE EXISTS                                                    *    2214
* 34     enddo UNTIL I IS REACHED                                                *    2215
*                                                                               *
*********************************************************************************
```

CALCULATE THE ADJACENT NODE'S TIME VALUES

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *  1    . AGAIN THIS IS SIMILAR TO THE TIME ROUTINE FOR THE ROAD NETWORK, AND  *   2217
     *  2    .AGAIN THE MAIN DIFFERENCE IS THE DATA BASE.                           *   2218
     *  3    .                                                                      *   2219
     *  4    CALCULATE THE TIME OF TRAVEL BETWEEN J AND THIS NODE                   *   2220
     *  5    CALCULATE CUMMULATIVE TIME FOR THIS LINK                               *   2221
     *  6    if THE CUMMULATIVE TIME CALCULATED IS LESS THAN THE WORKING LIST ENTRY *   2222
     *  7        SET NEW CUMMULATIVE TIME                                           *   2223
     *  8        SET NEW TIME MEASURE                                               *   2224
     *  9        SET PREDECESSOR NODE                                               *   2225
     * 10        SET CUMMULATIVE WORTH                                              *   2226
     * 11        SET WORTH MEASURE                                                  *   2227
     * 12    endif THE CUMMULATIVE TIME CALCULATED IS LESS                          *   2228
     *                                                                              *
     ********************************************************************************
```

CONSTRUCT SOLUTION←VECTORS AND ROUTE←VECTOR

```
REF
PAGE ***************************************************************************************
     *                                                                                     *
     *  1    . THIS ROUTINE IS IDENTICAL TO THE ROAD NETWORK ROUTINE WHICH PERFORMS        *    2230
     *  2    .THE SAME TASK EXCEPT THAT THE WORKING LIST IS REFERENCED IN ORDER TO         *    2231
     *  3    .OBTAIN CERTAIN VALUES, SUCH AS 1) TOTAL WORTH MEASURE, 2) STARTING           *    2232
     *  4    .TIME, 3) TIME MEASURE (LINK TIME), 4) PREDECESSOR'S CUMMULATIVE TIME,        *    2233
     *  5    .AND 5) THIS NODE'S CUMMULATIVE TIME                                          *    2234
     *                                                                                     *
     ***************************************************************************************
```

DISPLAY RESULTS

```
REF
PAGE ************************************************************************************
     *                                                                                  *
     *  1    do UPON FUNCTION REQUEST                                                   *  2236
     *  2        do FOREVER                                                             *  2237
     *  3            DISPLAY FUNCTION MENU                                              *  2238
     *  4            WAIT FOR USER INPUT                                                *  2239
     *  5            do CASE OF                                                         *  2240
     *  6 TABLE:                                                                        *  2241
  64 *  7                PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES        *  2242
     *  8 EXIT:                                                                         *  2243
     *  9                return TO CALLING PROGRAM                                      *  2244
     * 10            enddo CASE OF                                                     *  2245
     * 11        enddo FOREVER                                                         *  2246
     * 12    enddo UPON FUNCTION REQUEST                                               *  2247
     *                                                                                  *
     ************************************************************************************
```

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
		DI	MAIN←PROGRAM
			26 PREPARATION
			1
			35 LAMAS
			1 15
		DI	MAP←DIRECTORY
			26 PREPARATION
			2
			27 CREATE NODE←VECTOR AND MAP←DIRECTORY FILES ON DISK
			21 43 45
			32 CREATE MAP←DIRECTORY ENTRY FOR THIS MAP
			8 9
			39 ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS
			12 16 17 23
			40 READ THIS MAP'S NODES INTO MAIN MEMORY
			18 19 20
			41 CHANGE ADJACENT NODE'S MAP←NUMBER TO INDEX
			16
		DI	MAP←NUMBER
			32 CREATE MAP←DIRECTORY ENTRY FOR THIS MAP
			8
			33 CREATE CROSS-COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK
			50
			38 INITIALIZE MAP←NUMBERS AND SCREEN EXTREMES
			5 6 7
			39 ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS
			15 16 21 22 24 26 27
			41 CHANGE ADJACENT NODE'S MAP←NUMBER TO INDEX
			16
			43 ESTABLISH A FORWARD EDGE OF BATTLE AREA
			5 9 14

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
			46 PERFORM INTERDICTIVE OPERATIONS
			11
			47 PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINAT
			17
			48 CHANGE CONTENTS OF A NODE←VECTOR
			7 12
			50 DELETE A NODE FROM THE NODE←VECTOR ARRAY
			9
			51 PRINT CONTENTS OF A NODE←VECTOR
			2
			60 CALCULATE SECOND BEST PATH
			62
			62 CALCULATE BEST NODE AT WHICH TO INTERDICT
			76 77
			67 PERFORM TERRAIN INITIALIZATION
			10 11
			68 MAP←NUMBER TO BE USED
			8
		DI	MAP←NUMBERS
			37 PREPARE FOR ROAD NETWORK PATH CALCULATIONS
			7
			39 ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS
			13 14
		DI	NODE←NUMBER
			41 CHANGE ADJACENT NODE'S MAP←NUMBER TO INDEX
			16
			43 ESTABLISH A FORWARD EDGE OF BATTLE AREA
			5 9 14
			46 PERFORM INTERDICTIVE OPERATIONS
			11
			47 PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINAT
			17

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
			48 CHANGE CONTENTS OF A NODE←VECTOR
			7 12
			50 DELETE A NODE FROM THE NODE←VECTOR ARRAY
			9
			51 PRINT CONTENTS OF A NODE←VECTOR
			2
			60 CALCULATE SECOND BEST PATH
			62
			62 CALCULATE BEST NODE AT WHICH TO INTERDICT
			76 77
		DI	NODE←VECTOR
			26 PREPARATION
			2
			27 CREATE NODE←VECTOR AND MAP←DIRECTORY FILES ON DISK
			34 35 36 38 44
			35 LAMAS
			7
			43 ESTABLISH A FORWARD EDGE OF BATTLE AREA
			14 15
			44 FIND NODE←VECTOR NEAREST TO UTM COORDINATES
			9 12 15
			46 PERFORM INTERDICTIVE OPERATIONS
			11 13 15 17
			47 PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINAT
			7 11 13 17
			48 CHANGE CONTENTS OF A NODE←VECTOR
			8 11 13 16 17 17 38 38 39 40
			49 CHANGE CHARACTERISTIC
			7
			50 DELETE A NODE FROM THE NODE←VECTOR ARRAY
			9
			51 PRINT CONTENTS OF A NODE←VECTOR
			7

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
			53 CALCULATE PATH
			78 83
			54 PURGE ALL ROUTES
			13 15
			55 REINITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES
			4 14
			56 CALCULATE AND ASSIGN TIME AND WORTH VALUES
			33 35
			59 COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
			46 58
			60 CALCULATE SECOND BEST PATH
			42 57
			61 FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
			49
			62 CALCULATE BEST NODE AT WHICH TO INTERDICT
			68 71
		DI	NODE←VECTORS
			27 CREATE NODE←VECTOR AND MAP←DIRECTORY FILES ON DISK
			20
			38 INITIALIZE MAP←NUMBERS AND SCREEN EXTREMES
			2
			48 CHANGE CONTENTS OF A NODE←VECTOR
			2C
		DI	NUMBER←OF←MAPS
			32 CREATE MAP←DIRECTORY ENTRY FOR THIS MAP
			9
		DI	ROAD←NETWORK
			35 LAMAS
			6

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
		DI	ROUTE←VECTOR
			53 CALCULATE PATH
			15
			54 PURGE ALL ROUTES
			6 8
			60 CALCULATE SECOND BEST PATH
			35 59
			61 FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
			53 69
			62 CALCULATE BEST NODE AT WHICH TO INTERDICT
			41 70 73
			72 CALCULATE CROSS-COUNTRY PATH
			24
		DI	ROUTE←VECTORS
			61 FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
			66
			62 CALCULATE BEST NODE AT WHICH TO INTERDICT
			74
		DI	SOLUTION←VECTOR
			54 PURGE ALL ROUTES
			9 11 14
			56 CALCULATE AND ASSIGN TIME AND WORTH VALUES
			40 42 44 46 49 67
			59 COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
			24 25 46 47 47 48 53 54 58
			61 FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
			49
			62 CALCULATE BEST NODE AT WHICH TO INTERDICT
			48
			72 CALCULATE CROSS-COUNTRY PATH
			27

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
		DI	SOLUTION←VECTORS
			53 CALCULATE PATH
			15
			59 COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
			33 43 65
			60 CALCULATE SECOND BEST PATH
			35 59
			61 FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
			53 66 69
			62 CALCULATE BEST NODE AT WHICH TO INTERDICT
			41 70 73 74
			72 CALCULATE CROSS-COUNTRY PATH
			24 29
		DI	TERRAIN←MODEL
			35 LAMAS
			8
		DI	UNIT←COUNTER
			42 INITIALIZE UNIT←VECTORS
			9 14 26 58
			69 ESTABLISH UNIT←VECTORS
			11 14 48
		DI	UNIT←VECTOR
			42 INITIALIZE UNIT←VECTORS
			27 33 34 40 46 52
			52 CHANGE PROPERTIES OF A UNIT
			10
			60 CALCULATE SECOND BEST PATH
			60 61
			69 ESTABLISH UNIT←VECTORS
			35

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
		DI	UNIT←VECTORS
			37 PREPARE FOR ROAD NETWORK PATH CALCULATIONS
			9
			42 INITIALIZE UNIT←VECTORS
			8
			60 CALCULATE SECOND BEST PATH
			16 19
			62 CALCULATE BEST NODE AT WHICH TO INTERDICT
			22 25
			67 PERFORM TERRAIN INITIALIZATION
			13
			69 ESTABLISH UNIT←VECTORS
			13

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
56		FS	CALCULATE AND ASSIGN TIME AND WORTH VALUES
			53 CALCULATE PATH
			08
62		FS	CALCULATE BEST NODE AT WHICH TO INTERDICT
			45 PATH DETERMINATION AND DISPLAY
			32
72		FS	CALCULATE CROSS-COUNTRY PATH
			71 PERFORM TERRAIN PATH CALCULATIONS
			7
53		FS	CALCULATE PATH
			45 PATH DETERMINATION AND DISPLAY
			26
60		FS	CALCULATE SECOND BEST PATH
			45 PATH DETERMINATION AND DISPLAY
			28
76		FS	CALCULATE THE ADJACENT NODE'S TIME VALUES
			75 CALCULATE THE CROSS-COUNTRY PATH
			16
75		FS	CALCULATE THE CROSS-COUNTRY PATH
			72 CALCULATE CROSS-COUNTRY PATH
			19
41		FS	CHANGE ADJACENT NODE'S MAP+NUMBER TO INDEX
			38 INITIALIZE MAP+NUMBERS AND SCREEN EXTREMES
			6

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
49		FS	CHANGE CHARACTERISTIC
			48 CHANGE CONTENTS OF A NODE←VECTOR
			25 27 29 31 33 35
48		FS	CHANGE CONTENTS OF A NODE←VECTOR
			46 PERFORM INTERDICTIVE OPERATIONS
			13
70		FS	CHANGE ENTERED COORDINATES TO AN INDEX INTO THE DATA BASE
			69 ESTABLISH UNIT←VECTORS
			34
52		FS	CHANGE PROPERTIES OF A UNIT
			46 PERFORM INTERDICTIVE OPERATIONS
			19
57		FS	COMPUTE NEXT NODE TO LABEL
			53 CALCULATE PATH
			12
59		FS	COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
			53 CALCULATE PATH
			15
			60 CALCULATE SECOND BEST PATH
			35 59
			61 FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
			69
			62 CALCULATE BEST NODE AT WHICH TO INTERDICT
			41 70 73
77		FS	CONSTRUCT SOLUTION←VECTORS AND ROUTE←VECTOR
			72 CALCULATE CROSS-COUNTRY PATH
			24

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
33		FS	CREATE CROSS-COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK
			26 PREPARATION
			3
30		FS	CREATE LINK CARD
			27 CREATE NODE←VECTOR AND MAP←DIRECTORY FILES ON DISK
			10
32		FS	CREATE MAP←DIRECTORY ENTRY FOR THIS MAP
			27 CREATE NODE←VECTOR AND MAP←DIRECTORY FILES ON DISK
			43
28		FS	CREATE NODE CARD
			27 CREATE NODE←VECTOR AND MAP←DIRECTORY FILES ON DISK
			7
27		FS	CREATE NODE←VECTOR AND MAP←DIRECTORY FILES ON DISK
			26 PREPARATION
			2
50		FS	DELETE A NODE FROM THE NODE←VECTOR ARRAY
			46 PERFORM INTERDICTIVE OPERATIONS
			15
78		FS	DISPLAY RESULTS
			71 PERFORM TERRAIN PATH CALCULATIONS
			9
43		FS	ESTABLISH A FORWARD EDGE OF BATTLE AREA
			37 PREPARE FOR ROAD NETWORK PATH CALCULATIONS
			11

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
39		FS	ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE<VECTORS
			38 INITIALIZE MAP<NUMBERS AND SCREEN EXTREMES
			2
69		FS	ESTABLISH UNIT<VECTORS
			67 PERFORM TERRAIN INITIALIZATION
			13
44		FS	FIND NODE<VECTOR NEAREST TO UTM COORDINATES
61		FS	FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
			45 PATH DETERMINATION AND DISPLAY
			30
66		FS	IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE CROSS-COUNTRY DATA BASE
			35 LAMAS
			9
36		FS	IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE NODE<VECTOR DATA BASE
			35 LAMAS
			7
38		FS	INITIALIZE MAP<NUMBERS AND SCREEN EXTREMES
			37 PREPARE FOR ROAD NETWORK PATH CALCULATIONS
			7
42		FS	INITIALIZE UNIT<VECTORS
			37 PREPARE FOR ROAD NETWORK PATH CALCULATIONS
			9
35		FS	LAMAS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
63		FS	MAP←NUMBER TO BE USED
			67 PERFORM TERRAIN INITIALIZATION
			11
74		FS	OBTAIN THE PATH PARAMETERS
			72 CALCULATE CROSS-COUNTRY PATH
			15
73		FS	OBTAIN UNIT NAMES TO BE CONSIDERED
			72 CALCULATE CROSS-COUNTRY PATH
			13
45		FS	PATH DETERMINATION AND DISPLAY
			36 IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE NODE←VECTOR DATA BASE
			9
46		FS	PERFORM INTERDICTIVE OPERATIONS
			45 PATH DETERMINATION AND DISPLAY
			24
67		FS	PERFORM TERRAIN INITIALIZATION
			66 IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE CROSS-COUNTRY DATA BASE
			11
71		FS	PERFORM TERRAIN PATH CALCULATIONS
			66 IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE CROSS-COUNTRY DATA BASE
			13
26		FS	PREPARATION
37		FS	PREPARE FOR ROAD NETWORK PATH CALCULATIONS
			36 IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE NODE←VECTOR DATA BASE
			7

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
63		FS	PRESENT RESULTS
			45 PATH DETERMINATION AND DISPLAY
			34
51		FS	PRINT CONTENTS OF A NODE←VECTOR
			46 PERFORM INTERDICTIVE OPERATIONS
			17
47		FS	PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINATES
			46 PERFORM INTERDICTIVE OPERATIONS
			11
65		FS	PRINT PATH STATISTICS
			63 PRESENT RESULTS
			12
64		FS	PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES
			63 PRESENT RESULTS
			10
			78 DISPLAY RESULTS
			7
54		FS	PURGE ALL ROUTES
			53 CALCULATE PATH
			26
			50 CALCULATE SECOND BEST PATH
			10
			61 FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
			13
			62 CALCULATE BEST NODE AT WHICH TO INTERDICT
			16
			72 CALCULATE CROSS-COUNTRY PATH
			12

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
40		FS	READ THIS MAP'S NODES INTO MAIN MEMORY
			39 ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS
			20
55		FS	REINITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES
			53 CALCULATE PATH
			78 83
			60 CALCULATE SECOND BEST PATH
			42 57

SEGMENT REFERENCE TREES

PREPARATION

LN	DEF	SEGMENT
1	26	PREPARATION
2	27	CREATE NODE←VECTOR AND MAP←DIRECTORY FILES ON DISK
3	28	CREATE NODE CARD
4	30	CREATE ! INK CARD
5	32	CREATE MAP←DIRECTORY ENTRY FOR THIS MAP
6	33	CREATE CROSS-COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK

LAMAS

LN	DEF	SEGMENT
1	35	LAMAS
2	36	IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE NODE←VECTOR DATA BASE
3	37	PREPARE FOR ROAD NETWORK PATH CALCULATIONS
4	38	INITIALIZE MAP←NUMBERS AND SCREEN EXTREMES
5	39	ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS
6	40	READ THIS MAP'S NODES INTO MAIN MEMORY
7	41	CHANGE ADJACENT NODE'S MAP←NUMBER TO INDEX
8	42	INITIALIZE UNIT←VECTORS
9	43	ESTABLISH A FORWARD EDGE OF BATTLE AREA
10	45	PATH DETERMINATION AND DISPLAY
11	46	PERFORM INTERDICTIVE OPERATIONS
12	47	PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINATES
13	48	CHANGE CONTENTS OF A NODE←VECTOR
14	49	CHANGE CHARACTERISTIC
15	50	DELETE A NODE FROM THE NODE←VECTOR ARRAY
16	51	PRINT CONTENTS OF A NODE←VECTOR
17	52	CHANGE PROPERTIES OF A UNIT
18	53	CALCULATE PATH
19	54	PURGE ALL ROUTES
20	55	REINITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES
21	56	CALCULATE AND ASSIGN TIME AND WORTH VALUES
22	57	COMPUTE NEXT NODE TO LABEL
23	59	COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
24	60	CALCULATE SECOND BEST PATH
25	54	PURGE ALL ROUTES
26	59	COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
27	55	REINITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES
28	61	FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
29	54	PURGE ALL ROUTES
30	59	COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
31	62	CALCULATE BEST NODE AT WHICH TO INTERDICT
32	54	PURGE ALL ROUTES
33	59	COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
34	63	PRESENT RESULTS

LAMAS (CONTINUED)

LN	DEF	SEGMENT
35	64	PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES
36	65	PRINT PATH STATISTICS
37	66	IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE CROSS-COUNTRY DATA BASE
38	67	PERFORM TERRAIN INITIALIZATION
39	68	MAP←NUMBER TO BE USED
40	69	ESTABLISH UNIT←VECTORS
41	70	CHANGE ENTERED COORDINATES TO AN INDEX INTO THE DATA BASE
42	71	PERFORM TERRAIN PATH CALCULATIONS
43	72	CALCULATE CROSS-COUNTRY PATH
44	54	PURGE ALL ROUTES
45	73	OBTAIN UNIT NAMES TO BE CONSIDERED
46	74	OBTAIN THE PATH PARAMETERS
47	75	CALCULATE THE CROSS-COUNTRY PATH
48	76	CALCULATE THE ADJACENT NODE'S TIME VALUES
49	77	CONSTRUCT SOLUTION←VECTORS AND ROUTE←VECTOR
50	78	DISPLAY RESULTS
51	64	PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES