THE ENGINEER EFFECTS MODULE FOR THE STAR COMBAT MODEL (U)

CLASSIFIED

MAR 82 5 C MAIN. J V MUDD
THE ENGINEER EFFECTS MODULE FOR THE STAR COMBAT MODEL

by
Stephen Charles Main

and
James Vincent Mudd

March 1982

Thesis Advisor: James K. Hartman

Approved for public release: distribution unlimited
The Engineer Effects Module for the STAR Combat Model

Stephen Charles Main
James Vincent Mudd

Naval Postgraduate School
Monterey, California 93940

Naval Postgraduate School
Monterey, California 93940

Approved for public release: distribution unlimited

Minefield
STAR

Army
Obstacles

This thesis presents an Engineer Effects Module for the Simulation of Tactical Alternative Responses (STAR) combat model. The effects of engineer obstacles on the combat process are important, and the STAR combat model previously lacked the capability to model these effects. Implicit in the construction of the model is the task of modelling the obstacles followed by the simulation of the synergistic effects of the obstacles on the
The Engineer Effects Module is executed in the SIMSCRIPT II.5 programming language. This thesis serves as both the implementation and running instructions for the Engineer Effects Module for the STAR model.
Approved for public release: distribution unlimited

The Engineer Effects Module for the STAR Combat Model

by

Stephen C. Main
Captain, United States Army
B.S., United States Military Academy, 1972

and

James V. Mudd
Captain, United States Army
B.S., United States Military Academy, 1974

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March, 1982

Authors:

Approved by:

Thesis Advisor

Second Reader

Chairman, Department of Operations Research

Dean of Information and Policy Sciences
ABSTRACT

This thesis presents an Engineer Effects Module for the Simulation of Tactical Alternative Responses (STAR) combat model. The effects of engineer obstacles on the combat process are important, and the STAR combat model previously lacked the capability to model these effects. Implicit in the construction of the model is the task of modelling the obstacles followed by the simulation of the synergistic effects of the obstacles on the combat commander's actions. The Engineer Effects Module is executed in the SIMSCRIPT II.5 programming language. This thesis serves as both the implementation and running instructions for the Engineer Effects Module for the STAR model.
TABLE OF CONTENTS

I. INTRODUCTION - - - - - - - - - - - - - - - - - - - - - - - - - - - - 15

II. UNITED STATES TACTICS AND EQUIPMENT - - - - - - - - - - - - - 19
   A. OBSTACLES - - - - - - - - - - - - - - - - - - - - - - - - - - - - 21
      1. The Minefield - - - - - - - - - - - - - - - - - - - - - - - - - - - - 22
         a. Hasty Minelaying - - - - - - - - - - - - - - - - - - - - - - - - - - - 22
         b. Deliberate Minelaying - - - - - - - - - - - - - - - - - - - - - - - - - - - 22
         c. Special Minelaying - - - - - - - - - - - - - - - - - - - - - - - - - - - - 23
         d. Scatterable Minelaying - - - - - - - - - - - - - - - - - - - - - - - - - - - 23
      2. The Tank Ditch - - - - - - - - - - - - - - - - - - - - - - - - - - - - 24
      3. The Road Crater - - - - - - - - - - - - - - - - - - - - - - - - - - - - 24
      4. The Destroyed Bridge (short-wet gap) - - - - - - - - - - - - - - - 25
   B. TACTICAL RESPONSES TO OBSTACLES - - - - - - - - - - - - - - - 26
      1. The Bypass - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 26
      2. The Breach - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 26
      3. The Force Through - - - - - - - - - - - - - - - - - - - - - - - - - - - - 27
         a. Neutralizing and Breaching Minefields - - - - - - - - - - - - - - - 28
         b. Tank Ditches/Craters - - - - - - - - - - - - - - - - - - - - - - - - - - - 29
         c. Crossing Short-Wet Gaps - - - - - - - - - - - - - - - - - - - - - - - - - - 29

III. SOVIET TACTICS AND EQUIPMENT - - - - - - - - - - - - - - - - - - 30
   A. INTRODUCTION - - - - - - - - - - - - - - - - - - - - - - - - - - - - 30
   B. SOVIET FORCES AND EQUIPMENT - - - - - - - - - - - - - - - - - - - 33
C. THE MINEFIELD IN STAR - - - - - - - - - - - - - - - - - - 61

1. Minefield Entry Actions - - - - - - - - - - - - - - - - - - 62
   a. Routine MINE.SCHEDULE - - - - - - - - - - - - - - - - 62

2. Minefield Internal Actions- - - - - - - - - - - - - - - - - - 67
   a. Routine POP.A.MINE- - - - - - - - - - - - - - - - 67
   b. Routine PROB.MINE - - - - - - - - - - - - - - - - 74

3. Field Exit Actions- - - - - - - - - - - - - - - - - - - - 76

D. SYNERGISTIC EFFECTS - - - - - - - - - - - - - - - - - - - - 76

1. Tactical Alternatives for the Commander - - 76

2. Minefield Synergistic Actions - - - - - - - - - - - - - - - - 77
   a. Actions with No Prior Knowledge - - - - - - - - - - - 80
   b. Actions with Knowledge Prior to Entry - - 84
   c. Actions When a Lane is Known to Exist - - 84

3. The Decision Ellipse - - - - - - - - - - - - - - - - - 86

E. DETAILED FLOWCHART APPROACH - - - - - - - - - - - - - - - 92

1. The Decision Ellipse (Detailed) - - - - - - - - - - - 94

2. The Minefield Ellipse - - - - - - - - - - - - - - - - - - - 97
   a. Detailed Minefield Entry Actions- - - - - - - - - - 97
   b. Detailed Minefield Internal Actions - - - - 103
   c. Detailed Minefield Exit Actions - - - - - - - - - - - 109

F. CONCLUSION OF THE MINEFIELD MODEL - - - - - - - - - - - - - - 111

VI. THE ENGINEER GAP OBSTACLES- - - - - - - - - - - - - - - - - - 112

A. INTRODUCTION- - - - - - - - - - - - - - - - - - - - - - - - 112
B. GAP OBSTACLES MODELLED

1. The Tank Ditch
2. The Road Crater
3. The Blown Bridge (short-vet gap)

C. THE GAP OBSTACLE IN STAR

1. Gap Decision Ellipse Actions
2. Gap Entry Actions
3. Event GAP.JOCK
4. Gap Internal Actions
5. Event HEAVY.JUNK
6. Event GAP.BREACH
7. Gap Exit Actions

VII. FURTHER ENHANCEMENTS

A. FURTHER CAPABILITIES OF THE OBSTACLES REPRESENTED

B. ENGINEER UNIT REPRESENTATION IN STAR

1. Engineers in the Offense
2. Engineers in the Defense

C. EQUIPMENT ENHANCEMENTS/CAPABILITIES

D. ADDITIONAL OBSTACLE REPRESENTATION

E. OTHER TACTICAL OPTIONS

F. MISCELLANEOUS

APPENDIX A: EXPLANATION OF THE ARRAY TRAP CONTROL
A. THE DIMENSIONING OF TRAP.CONTROL - 148
B. THE CONTENTS OF TRAP.CONTROL - 150
   1. Minefield TRAP.CONTROL Contents - 151
   2. Gap Obstacle TRAP.CONTROL Contents - 152
C. THE FUNCTION OF TRAP.CONTROL - 153
   1. TRAP.CONTROL Function in Minefields - 153
   2. TRAP.CONTROL Functions In Gap Obstacles - 157

APPENDIX B: MODIFIED ROUTINES AND EVENTS FROM BASIC

STAR - 159
1. PREAMBLE - 159
2. Routine BLCREATE - 167
3. Routine MOVE - 169
4. Routine MOVE.LIMITS - 171
5. Routine REDCREATE - 172
6. Routine RES2 - 173
7. Routine RES5 - 174
8. Event TARGET.SELECT - 175
9. Routine FLD.ACT - 177
10. Routine FLD.CREATE - 180
11. Routine FLD.INIT - 182
12. Routine BASIC.LOAD - 184

APPENDIX C: ROUTINES AND EVENTS FOR THE ENGINEER MODULE

IN STAR - 186
1. Routine POP.A.MINE - 186
2. Routine FIND.A.MINE - 200
3. Routine PROB.MINE - 205
4. Routine MF.DECISION - 210
5. Routine MF.ENTRY - 219
6. Routine DROP.FLOW - 230
7. Routine WITHDRAW - 234
8. Routine MF.INTERNAL - 237
9. Routine MF.EXIT - 250
10. Event HALT - 255
11. Event TURN.AROUND - 257
12. Event STAND.TO - 259
13. Event DIVERT - 261
14. Routine MOMENTUM - 264
15. Routine PLW.ALIVE - 267
16. Event QUIK.MOVE - 270
17. Routine MINE.SCHED - 273
18. Routine GAP.DECISION - 282
19. Routine JUNK.ALIVE - 292
20. Routine GAP.ENTRY - 295
21. Routine GAP.INTERNAL - 304
22. Routine GAP.LEAVE - 313
23. Event DITCH.KILL - 317
APPENDIX D: USER INPUT INSTRUCTIONS FOR THE STAR ENGINEER MODULE

A. INTRODUCTION

B. OBSTACLE/FIELD REPRESENTATION
   1. Minefields
      a. Input Example for a Patterned Minefield
      b. Input Example for a Scatterable Minefield
   2. Gap Obstacles
   3. Decision Ellipses
      a. Input Example for a Minefield Decision Ellipse

C. ARMORED VEHICLE LAUNCHED BRIDGE REPRESENTATION

D. OFFSET FORMATIONS

E. TANK BREACHING ATTACHMENTS

LIST OF REFERENCES

INITIAL DISTRIBUTION LIST
LIST OF TABLES

1. Soviet Army Amphibious Vehicle Capabilities - - - - 34
2. Organic Breaching Equipment of a Soviet Regiment- - 34
3. Organic Breaching Equipment of a Soviet Division- - 35
4. Organic Breaching Equipment of a Soviet Front/Army- 36
5. Tank-mounted Mine Clearing Equipment - - - - - - - - 36
6. GRUNTLETH - - - - - - - - - - - - - - - - - - - - - - - - 74
7. MINELETH - - - - - - - - - - - - - - - - - - - - - - - - 75
8. Tactical Alternatives at an Obstacle Encounter- - 78
9. Decision Ellipse Attributes - - - - - - - - - - - - - - - - - 94
10. Decision Ellipse Tactical Alternatives- - - - - - - - - - 96
11. Minefield Ellipse Attributes- - - - - - - - - - - - - - - - 97
LIST OF FIGURES

1. Minefield Representation - - - - - - - - - - - 54
2. Representation of fragmentation mine - - - - - - - 58
3. Effective Tripwire Length - - - - - - - - - - - - 59
4. Claymore Mine Representation - - - - - - - - - - 60
5. Routine MINE.SCHEDULE Flowchart - - - - - - - - 63
6. Routine POP.A.MINE Flowchart - - - - - - - - - - 68
7. Routine PROB.MINE Flowchart - - - - - - - - - - - 73
8. General Minefield Boundary Actions Flowchart - - - 79
9. General Decision Ellipse Flowchart - - - - - - - - - 87
10. Decision Ellipse Example - - - - - - - - - - - - - 89
11. Minefield Decision Ellipse Flowchart - - - - - - - 93
12. Minefield Entry Action Flowchart - - - - - - - - - 98
13. Minefield Internal Actions Flowchart - - - - - - - 104
14. Minefield Exit Actions Flowchart - - - - - - - - - 110
15. Tank ditch representation - - - - - - - - - - - - 113
16. Road Crater/Short-Wet Gap Representation - - - - - 114
17. Routine GAP.DECISION Flowchart - - - - - - - - - 118
18. Routine GAP.ENTRY Flowchart - - - - - - - - - - - 121
19. Event GAP.JOCK Flowchart - - - - - - - - - - - - 124
20. Routine GAP.INTERNAL Flowchart - - - - - - - - - 126
21. Event WALL.BREACH Flowchart - - - - - - - - - - - 129
22. Event HEAVY.JUNK Flowchart - - - - - - - - - - - - 132
23. Multiple Breach of a Tank Ditch, Time: 9:00 - - - 134
24. Multiple Breach of a Tank Ditch, Time: 9:02 - - - 135
25. Event GAP.BREACH Flowchart - - - - - - - - 136
26. Routine GAP.LEAVE Flowchart - - - - - - - - - 139
I. INTRODUCTION

The Simulation of Tactical Alternative Responses (STAR) combat model, as it currently exists, does not model Engineer obstacles or counter-obstacle operations. It is the purpose of the Engineer Effects Module to install this capability in the STAR Combat model. This Engineer Effects Module consists of the representation of four types of obstacles and their effects on the battlefield processes. The obstacles are the minefield, tank ditch, road crater, and the destroyed bridge (short-wet gap). The minefield actually consists of the modelling of five different types of fields, the anti-tank patterned minefield, the anti-tank scatterable minefield, the anti-personnel blast minefield, the anti-personnel fragmentation minefield and the Claymore mine.

The obstacles are important in their own right, however, the synergistic effects of the obstacles on the forces and tactics being employed are the crucial effects to be modelled. The modelling of land combat and the ability to capture synergistic effects on the battlefield are difficult and complex tasks.
In order to model the effects of the Engineer obstacles mentioned above, the counter-obstacle tactics for the combat forces had to be developed and modelled. The tactics that are modelled for the obstacles are as follows:

1. For the minefield:
   - Conduct a hasty breach with plows.
   - Conduct a bypass.
   - Bull through the field.

2. For tank ditches and road craters:
   - Conduct a crossing of the gap with an Armored Vehicle Launched Bridge
   - Conduct a lane filling operation using tanks equipped with dozer blades.
   - Conduct a self-breaching operation by allowing each vehicle to enter the obstacle and create a breach by caving the obstacle walls in.
   - Conduct a bypass of the obstacle, when no proper suit is present and the risk of trying a self breach is too great.

3. For the short-wet gap
   - Conduct a hasty crossing using the Armored Vehicle Launched Bridge (AVLB).
   - Conduct a bypass of the gap if there is no AVLB.

The capability to breach obstacles necessitated the introduction of a new system in STAR, the armored vehicle launched bridge (AVLB), and the expansion of the capability of existing systems through the representation of mine plows and bull dozer blades.
In Chapter 2, the current capabilities and equipment of the United States Army are discussed. This information forms the basis for the effects that are modelled for the United States and NATO forces.

In Chapter 3, the capabilities and equipment of the forces of the Soviet Union are described and discussed. This chapter forms the basis for the tactics that are modelled for the Warsaw Pact forces in STAR.

Chapter 4 consists of a review of some of the past modelling efforts and ideas that exist in the Engineer arena.

Chapter 5 consists of a detailed description of the minefields and their associated tactical responses constructed for the STAR combat model.

Chapter 6 is the detailed description of the remaining obstacles (tank ditch, road crater, and destroyed bridge) and their synergistic effects.

Finally, Chapter 7 consists of some enhancements the authors see as possibilities for the model in the future. Modeling is viewed by the authors as an iterative process and this chapter may assist in the future development of the model. In the appendices, variables are defined, changes to
existing STAR code are explained, and the computer code is listed and described in detail. Complete running instructions are also included.

It is the intention of the authors that this Engineer effects module can be fully implemented by a serious user of the STAR combat model using only this document and the existing STAR documentation.
II. UNITED STATES TACTICS AND EQUIPMENT

History has shown that obstacles and barrier systems are an effective casualty producing ingredient on the battlefield. Most obstacles used were minefields or obstacles reinforced with mines. The following table lists Allied tank losses to mines as a percentage of losses from all enemy action [Ref. 1].

<table>
<thead>
<tr>
<th>THEATER</th>
<th>PERCENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>North Africa 1942-43</td>
<td>18</td>
</tr>
<tr>
<td>Western Europe 1944-45</td>
<td>23</td>
</tr>
<tr>
<td>Italy 1943-45</td>
<td>28</td>
</tr>
<tr>
<td>Pacific 1944-45</td>
<td>34</td>
</tr>
<tr>
<td>Korea 1950-55</td>
<td>56</td>
</tr>
<tr>
<td>Vietnam 1967-69</td>
<td>69</td>
</tr>
</tbody>
</table>

More recently, obstacle use played a major role in the Mid-East War of October 1973. On the plains of the Golan Heights, the Israelis constructed an anti-tank ditch approximately 15 kilometers long. The ditch was heavily reinforced with minefields on both the enemy and the friendly sides. The combination of the Israelis' decisive use of the terrain/terrain reinforcement and mines with covering fire allowed their grossly outnumbered force of 180 tanks, 11 batteries of Artillery, and supporting Engineers to defeat an attacking Syrian force of 800 tanks, 115
batteries of Artillery, and Engineers (Ref. 2). The above historical evidence highlights the previous use of obstacles on battlefields of the past. The evidence should spark an interest in how their use will affect the battlefield of today and tomorrow.

The STAR model, as it currently exists, does not model any Engineer obstacles. STAR does however, possess the inherent capability to do so using the field module with some major modifications (Ref. 3). This chapter will briefly describe the following items as a prelude to attempting to model Engineer effects in STAR.

1. Four general obstacles:
   - the minefield
   - the road crater
   - the tank ditch
   - the destroyed bridge (short-gap)

2. The alternatives available to the tactical commander in order to overcome an obstacle in his path of advance.

In this model, the actual employment of the Engineers and their equipment will not be discussed. If the reader wishes to investigate this area more deeply, he should refer to References 4, 5, and 6. It is acknowledged that the
combat Engineer is a viable member of the combined arms team. Engineers increase the combat effectiveness of the combat forces on the battlefield. However, in order to model the synergistic effects of Engineer obstacles in STAR, the existing model must form the basis of the iterative process. After the obstacles and their effects are modelled, then the Engineer units may be added to the battle being played. In short, the first step in the process is to install an Engineer Obstacle Effects Module in STAR.

A. OBSTACLES

An obstacle is any obstruction that stops, delays, or restricts movement. Obstacles have been classified as natural/existing or artificial/reinforcing/manmade. The primary purposes of obstacles are to enhance the effectiveness of friendly fires, to delay and disrupt the enemy formations, to divert the enemy, to allow the tactical commander to use economy of force and to enable him to protect his flanks. Because threat tactics are based on the massive use of mechanized forces, one could say that the principal purpose of battlefield obstacles is to overcome the inherent mobility advantage of the enemy's tank forces [Ref. 7]. For the purpose of this discussion, only the four
major anti-armor obstacles previously mentioned will be discussed.

1. The Minefield

Minefields are located where they will delay armor and strengthen the defense, but they are not used with the idea of blocking or destroying the enemy force. Mines reinforce natural obstacles, scare the enemy, cause his attention to be diverted from the defender, and influence his maneuver. [Ref. 8] For the purpose of model building, the names of the various types of minefields are not important but the methods of installation are. There are four basic techniques used in laying minefields: [Ref. 9]

a. Hasty Minelaying

Hasty minelaying is done near the forward edge of the battlefield without Engineer support. Mines are emplaced so that friendly troops may detect and recover the mines. Usually, they are laid on top of the ground in a random fashion or in a specific pattern, i.e. a row.

b. Deliberate Minelaying

Deliberate minelaying normally follows a pattern. Each mine is buried and camouflaged. Mine clusters and strips are plotted and marked. Boundaries are fenced
and marked. This type of minelaying is normally done by the Engineers and is used when there is time to plan, organize, and prepare the logistical support for the effort.

c. Special Minelaying

Special minelaying is the mining of railroads, routes, stream beds, and other unique point targets on the battlefield. It is normal to improvise when laying mines for these purposes.

d. Scatterable Minelaying

Mines can be delivered by artillery, aircraft, missiles, ground dispensers, or thrown by hand. This method allows the commander to have a rapid minefield installation capability that is not manpower intensive.

The other three categories of obstacles can be thought of as anti-vehicular obstacles. Anti-vehicular obstacles should not be continuous across the front of a position, but should have gaps which can be kept under observation and covered by fire. Such gaps tend to channelize the enemy's movement. These obstacles are located to take advantage of natural concealment for a surprise effect and are usually reinforced with mines. [Ref. 10]
2. **The Tank Ditch**

The tank ditch is a linear obstacle emplaced normally by mechanical digging machines. There are two primary types of ditches, the triangular ditch and the rectangular ditch. The triangular ditch has a 1.5 to 1.8 meter vertical wall with berm and the rectangular ditch is 3.3 meters wide, 1.5 to 1.8 meters deep and also has a berm. The vertical wall or depth varies because of different soil types. The triangular ditch does not present an obstacle to the friendly or counter-attacking force. [Ref. 2] The rectangular ditch presents an obstacle to both the attacking force and the friendly force.

3. **The Road Crater**

Crater type obstacles are used for blocking roads, trails, or defiles. The location of the crater is chosen such that no bypass exists or if one exists, the bypass route can either be mined, covered by friendly fire or both. Craters are formed by explosive charges installed and prepared in advance for later detonation. [Ref. 10] Road Craters, in order to be effective obstacles, must be too wide to be spanned by the natural bridging capability of tracked vehicles. In addition, they must be deep and steep-
sided to prevent vehicles from passing through them. Road

craters created by blast will not stop modern tanks

indefinitely, but are effective obstacles if the tank

requires three or more passes in order to cross the crater.

Three passes provide sufficient time for the anti-tank

weapon to destroy a tank by fire. [Ref. 11]

a. The Destroyed Bridge (short-wet gap)

Generally, bridges are destroyed to create obstacles

which delay the enemy. However, bridges seldom require

complete destruction. The method used for demolition should

normally permit the economical reconstruction of the bridge

by friendly troops in future operations [Ref. 11].

Now that the four basic obstacles have been

described, one may ask the question, What makes an obstacle

effective? An effective obstacle is one which slows and

delays the enemy and is covered by fire. This is to counter

enemy efforts to breach and/or destroy the enemy while

entrapped by the obstacle. Obstacles should be employed in

such a way as to surprise the enemy and take advantage of

his confusion. Finally, they should be of no advantage to

the enemy, any cover and concealment provided by the

obstacle should be mined or booby-trapped. [Ref. 8]
B. TACTICAL RESPONSES TO OBSTACLES

The enemy force has the same capabilities and opportunities to emplace obstacles as the friendly force. Therefore, it is important to discuss the friendly tactical commander's possible actions upon encountering an obstacle on the battlefield. The friendly commander must keep one principle in mind when an obstacle is encountered, maintain the momentum of the offense. Obstacles must not impede the movement for unusually long periods of time. The commander has three alternatives when faced with an obstacle; he may bypass, "force through" or breach the obstacle. [Ref. 7]

1. The Bypass

To conserve time and manpower, obstacles are bypassed whenever possible. However, if the enemy has employed these obstacles cleverly, they will be difficult to bypass [Ref. 7]. The defender has probably covered all bypass routes by fire. The unit should conduct the bypass with attention to cover, concealment, and suppression. [Ref. 12]

2. The Breach

If bypassing is not possible, then another choice for the commander is to attempt a breach. A breach is
conducted when the unit possesses the proper equipment to breach. Two methods of breaching may be employed, the assault breach or the deliberate breach. [Ref. 7]

a. The assault/hasty breach is done quickly during either hasty or deliberate attacks. This is a tactical breach. The assault troops are under enemy direct or indirect fire. The main objective is speed in gaining the breach. Delays while breaching, are more costly in terms of causing casualties than the mines.

b. The deliberate breach is conducted when enemy fires covering the field have been neutralized and safety is more important than time. The performance of a deliberate breach is an Engineer responsibility. Because Engineer units are not currently modelled in STAR, this tactic will not be discussed as an alternative for the tactical commander.

3. **The Force Through**

   The "force/bull through" is attempted when no other way to overcome the obstacle is possible. The unit does not possess any breaching equipment and will just drive their vehicles through the minefield hoping to get through and perhaps clear a path. Heavy losses are expected when this tactic is employed. [Ref. 12]
The breach options demand further explanation. The tactical commander should breach obstacles by applying these fundamentals: [Ref. 7]

1. Suppress enemy weapons using available fire.
2. Use smoke to obscure enemy observation.
3. Secure the far side of the obstacle.
4. Reduce the obstacle.

The fourth fundamental is of specific interest in order to model obstacles in STAR. How does the tactical commander reduce obstacles, or to be more specific, how does he combat the obstacles discussed in Section A of this chapter?

a. Neutralizing and Breaching Minefields

The use of mechanical or explosive breaching devices is preferred over manual methods unless a special need exists. [Ref. 7] Explosive devices require Engineer support and for this reason are not discussed. The tactical commander therefore has only a tank mounted dozer blade (one per tank company) that can breach small point minefields or the Europe-deployed tank mounted mine roller for larger minefields [Ref. 6].
b. Tank Ditches/Craters

A tank dozer can be used to push down the sides of ditches or to fill craters in. If the gap created is 57 feet or less then the AVLB can be used. [Ref. 12] It should be noted that tank ditches and road craters are usually reinforced with mines and may require additional breaching equipment.

c. Crossing Short-Wet Gaps

Short gaps created by streams, small rivers, or valleys can be crossed using the existing civilian fixed bridge structures. However, if the enemy has destroyed these, or there were none in existence, then the tactical commander must resort to his organic assets. Normally, the AVLBs are utilized to cross gaps of less than 57 feet (17.4 meters) with vehicles of up to 60 tons in weight [Ref. 6].

This concludes the discussion of obstacles and tactical responses available to a commander of U.S. forces. The tactics and capabilities described in this chapter form the basis for modelling Engineer effects on Blue forces currently in the STAR model. The next chapter will discuss the capabilities and tactical alternatives of the Soviet Army.
III. SOVIET TACTICS AND EQUIPMENT

A. INTRODUCTION

This chapter is a consolidation of information on the Soviet forces and their capabilities in the areas of counter-obstacle operations and equipment. All the information presented here is unclassified and available in open literature. It is a widely accepted fact that Soviet military operations and tactics are directed towards the offensive. It is for this reason, that Soviet defensive operations will not be discussed. The primary mission of the Soviet Engineer is to enhance the mobility of the Soviet ground combat forces by countering obstacles to movement, be they natural or manmade. To construct an Engineer Effects Module for the Simulation of Tactical Alternative Responses (STAR) combat model one must determine the tactics for the Red forces. The research for this chapter provided the bulk of the tactical and equipment information required for the implementation of an engineer effects module for STAR.

The Soviet Union is preparing itself for a massive Air-Land battle with the forces of NATO. It has long been
hypothesized that the Warsaw Pact, led by the forces of the Soviet Union, will eventually attack west across the borders of Czechoslovakia and the German Democratic Republic into Western Europe. In order to be successful in their attack, the Warsaw Pact forces will rely on massive combined arms forces attacking on several avenues of advance with great speed. The NATO forces, on the other hand, will be defending with smaller forces. It is for this reason that the NATO forces will rely heavily on terrain reinforcement to slow the advancing Pact forces. These reinforcements will include minefields, tank ditches, road craters, blown bridges and other engineer obstacles. The Soviet Union is aware of this and consequently has amassed a rather impressive array of counter-obstacle forces and equipment. Further, in the Great Patriotic War the Soviet Union used many minefields and obstacles as terrain reinforcements against the advancing forces of Nazi Germany. [Ref. 13] In that war, the Soviet Union gave Nazi Germany a lesson on the effects of obstacles on the movement of ground combat forces. That lesson has not been forgotten by the teacher. Most of the high Soviet leadership of today fought in that war and thus, remembers the experience. The Soviet Union
not only is preparing to attack through the NATO defenses, but believes that it can be done successfully.

The Soviets developed into first-rate users of mine and obstacle warfare during the Great Patriotic War. It has been said that the battle of Kursk is the finest example of mine warfare in history. In that battle, some 3 million mines were laid to stop the Germans. In order to avert a transportation problem caused by the heavy mine usage, every time a soldier was sent to the front, he was required to take two mines with him. [Ref. 13, 14] The lessons learned as great users of obstacles have helped the Soviet Union to develop an arsenal of counter-obstacle equipment unparalleled in the world today.

It is interesting to note that the Soviet Union and the Warsaw Pact claim to be defensive forces, yet possess an offensive capability that is in a word, awesome. The NATO forces, though claimed to be the potential aggressors, possess a counter-obstacle capability that pales in comparison to that of the Warsaw Pact. As an example, the total mine roller capability of the US Army in Europe consists of 10 rollers that were scheduled for delivery to USAREUR in late 1981 [Ref 15]. The Soviet forces possess 9
rollers in just one Engineer Company supporting a hasty minefield breach of the forward combat elements [Ref. 13].

B. SOVIET FORCES AND EQUIPMENT

The Soviet Union possesses an impressive array of counter-obstacle equipment and forces in order to enhance the mobility of the Soviet combat arms. This description will focus on two primary areas:

1. The organic counter-obstacle elements in the ground combat arms.

2. The Engineer units organic to the larger units.

Perhaps the most obvious and least mentioned item is the design characteristics of the Soviet armored vehicles. Most of these vehicles are amphibious so that many water obstacles may be overcome easily with no additional bridging support. Although bridging equipment may be desirable, in many cases it is faster to just drive up to the river bank, enter the water, and cross. Most of the tanks in the Soviet army have an inherent snorkeling capability that allows them to cross water obstacles as deep as 5.5 meters at speeds of 2 km/hr [Ref. 15]. Table 1 is a consolidated list of the amphibious capability of some Soviet vehicles.
Table 1: Soviet Army Amphibious Vehicle Capabilities

<table>
<thead>
<tr>
<th>Vehicle/system</th>
<th>Water speed</th>
<th>Additional attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTR 50/60</td>
<td>10 km/hr</td>
<td></td>
</tr>
<tr>
<td>BMP</td>
<td>6 km/hr</td>
<td>can fire 73 mm main gun and Sagger while swimming</td>
</tr>
<tr>
<td>PT 76</td>
<td>10 km/hr</td>
<td>can fire 76 mm main gun while swimming</td>
</tr>
<tr>
<td>BRDM</td>
<td>10 km/hr</td>
<td></td>
</tr>
<tr>
<td>122mm SP HOW</td>
<td>approx 10 km/hr</td>
<td></td>
</tr>
<tr>
<td>SA-8/SA-9</td>
<td>approx 10 km/hr</td>
<td></td>
</tr>
</tbody>
</table>

[Ref. 16,17]

The above organic counter-obstacle capability is further supplemented by bridging assets at the Regimental level. These bridges can be used to cross water obstacles with no bridges left intact and may also be used to counter tank ditches and road craters. The following is a consolidation of the assets organic to the Motorized Rifle Regiment (MRR) and the Tank Regiment (TR).

Table 2: Organic Breaching Equipment of a Soviet Regiment

<table>
<thead>
<tr>
<th>Vehicle/system</th>
<th>Basis of issue</th>
<th>Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTU-20 (tank std bridge)</td>
<td>3 per Regt</td>
<td>12.3 to 20 meters</td>
</tr>
<tr>
<td>TMB (truck launched br)</td>
<td>4 per Regt</td>
<td>10.5 meters ea or 142 meters long</td>
</tr>
<tr>
<td>BTU Dozer Blades</td>
<td>3 per Regt</td>
<td>100-300 meters per hr by 3.4 meters wide</td>
</tr>
</tbody>
</table>

[Ref. 16,17,18]
In addition to these capabilities at the Regimental level, other assets exist in the Divisional Engineer Battalion to aid in counter-obstacle operations. They include the following:

Table 3: Organic Breaching Equipment of a Soviet Division

<table>
<thead>
<tr>
<th>Vehicle/system</th>
<th>Basis of issue</th>
<th>Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTU Dozer Blades</td>
<td>3 per div bn</td>
<td>100-300 meters per hr by 3.4 meters wide</td>
</tr>
<tr>
<td>HTU-20</td>
<td>4 per div bn</td>
<td>12.3 to 20 meters</td>
</tr>
<tr>
<td>THM</td>
<td>8 per div bn</td>
<td>10.5 meters ea or combined together.</td>
</tr>
<tr>
<td>PTS(tracked amphibian)</td>
<td>14 per div bn</td>
<td>carries 15000 kg of cargo at 15 km/hr</td>
</tr>
<tr>
<td>GSP(Ferry)</td>
<td>8 per div bn BD</td>
<td>4 50 metric ton ferries</td>
</tr>
<tr>
<td></td>
<td>12 per div bn TD</td>
<td>6 50 metric ton ferries</td>
</tr>
<tr>
<td>PNP(pontoon bridge)</td>
<td>18 per div bn</td>
<td>119 meters of 60 ton capacity bridge</td>
</tr>
</tbody>
</table>

[Ref. 16,17,18]

The Front and Army levels each have additional quantities of primarily the same equipment already mentioned. Table 4 lists the additional quantities at the Front or Army levels.

The man-made obstacles that will undoubtedly pose the greatest problems for the Soviet Commander are minefields. At the unit level, the mine-plow will be the primary asset, and it will be available in great numbers. The estimates vary, however it appears that the minimum number will be one
Table 4: Organic Breaching Equipment of a Soviet Front/Army

<table>
<thead>
<tr>
<th>Vehicle/system</th>
<th>Quantity</th>
<th>Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTU Dozer Blades</td>
<td>9</td>
<td>100-300 meters per hr by 3.4 meters wide</td>
</tr>
<tr>
<td>MTU-20</td>
<td>24</td>
<td>12.3-20 meters</td>
</tr>
<tr>
<td>THEM</td>
<td>48</td>
<td>10.5 meter ea</td>
</tr>
<tr>
<td>PMP</td>
<td>108</td>
<td>681 meters of 60 ton capacity bridge</td>
</tr>
<tr>
<td>PTS</td>
<td>98</td>
<td>15000 kg ea cargo</td>
</tr>
<tr>
<td>GSP</td>
<td>48</td>
<td>up to 110 tons at 7-8 km/hr</td>
</tr>
</tbody>
</table>

[Ref. 16,17,18]

plow per platoon, i.e. at least one tank in three will be equipped with a mine plow. However, normally each forward platoon has two tanks fitted with mine plows. [Ref. 19]

Every T-72 has the capability to mount a mine plow. [Ref. 16] The following is a table of the mine clearing equipment mounted on tanks:

Table 5: Tank-mounted Mine Clearing Equipment

<table>
<thead>
<tr>
<th>System</th>
<th>Basis of issue</th>
<th>Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>KMT4/6 Plow</td>
<td>3 per tank Co</td>
<td>10 km/hr 35 cm deep</td>
</tr>
<tr>
<td>PTS4-M Roller</td>
<td>3 per Regt and 3 per Eng Bn</td>
<td>10 km/hr</td>
</tr>
<tr>
<td>KMT-5 Plow/Roller Combination</td>
<td>4 per Regt</td>
<td>both of the above</td>
</tr>
</tbody>
</table>

[Ref. 16,17,18]

Along with the mine clearing equipment listed above the Soviet Union possesses many other items in the inventory to
deal with mines. Some of the items include, hand-held mine
detectors, bangalore torpedoes, explosive line charges, and
rocket propelled line charges. It is rather apparent that
the Soviet Union takes the necessity for counter-obstacle
capability very seriously. It is with this myriad of
equipment, employed in effective combined arms operations
that the Soviets plan to breach the NATO defenses in Western
Europe.

C. EMPLOYMENT OF SOVIET FORCES

"The offensive is the basic form of combat action. Only by a resolute offensive conducted at a high tempo
and to a great depth is the total destruction of the enemy attained."

General V.G. Reznichenko "Tactics"
[Ref. 17: p. 3-1]

In the previous section, the vast quantities of Soviet
counter-obstacle equipment were discussed. A discussion of
the tactics that the Soviets are expected to utilize in
order to accomplish their desired goals is contained herein.
A key aspect in the employment of the Soviet forces is to
capitalize on the high speeds of advance made possible with
mechanized forces. The mission of the Combined Arms Army or
Tank Army is to drive swiftly and deeply to the NATO Corps
rear boundary. [Ref. 17] If there is one dictum of the
Soviet Forces it is to maintain the speed of the advance.
Current military readings indicate that the Soviet commander desires to maintain a speed of advance of 50 to 80 kilometers per day under nuclear conditions and 40 to 50 kilometers per day under non-nuclear conditions. In order to be able to maintain these speeds, it will be necessary to reduce the effects of obstacles so that they do not hinder the advance of the Soviet forces. The Soviet commanders will be able to do this with their forces because of two primary reasons:

1. Large engineer units are located well forward with the combat arms.

2. Many simple engineer tasks will be handled by the ground gaining combat arms.

The Soviet forces will probably attack on high speed avenues of approach with one Motorized Rifle Regiment (MRR) in the lead followed by two additional regiments in the following echelon. The forces will attack with a high rate of advance and will usually pass pockets of resistance in order to maintain the high rate of advance.

The Soviet commander feels that in order to maintain a successful offensive operation, a numerical advantage of 3 to 1 is necessary. A Front will generally be utilized for the strategic offensive. This war-time only structure
usually consists of three or four Combined Arms Armies and one or two Tank Armies. The Front will attack with a structure similar to the following:

- **First Echelon** Three Combined Arms Armies (CAA)
- **Second Echelon** One CAA or One Tank Army
- **Front Reserve** One Tank or Motorized Rifle Division

The objectives of the Front are of three types, Immediate, Long Range, and Final. The definition of these are as follows:

- **Immediate** 250-280 km 3-5 days
- **Long Range** 500-560 km 7-13 days
- **Final** 1000 km 15-21 days

With the strategic view as discussed above, the discussion will now switch to the smaller unit levels for a look at the tactics. It is important to keep in mind the Soviet desire to maintain the speed of the attack. In light of this, as mentioned earlier, normally each forward platoon will have two tanks fitted with mine plows. This allows the platoon to make two parallel breaches simultaneously. [Ref. 19] As the lead reconnaissance elements move towards the NATO defenses, the elements will find the minefields either by detonating mines, visually, or with vehicle mounted mine
detectors. Depending on the situation, the elements may mark the minefields or they may clear lanes with the organic mine-plows and rollers. Engineer personnel may affix hose charges to the mine-plow vehicles to pull through the minefields. When the hose charge is detonated, the area between the cleared tracks from the plow or roller is also cleared of mines.

If the reconnaissance elements do not locate the minefields, then the follow-on units may. They will probably locate them by detonating a mine. At this point the commander has basically three courses of action. They are:

1. Divert off the desired axis of advance around the minefield.
2. Conduct a hasty breach with organic plows.
3. "Bull through" with whatever assets are available.

When the Soviet commander has Engineer units with him, then the most probable option will be to "storm" the minefield with the engineers in the lead. This operation is a combined arms operation with tanks providing direct fire, artillery providing indirect fire and smoke, and the organic plows clearing lanes under engineer supervision. The standard method is to use explosives to augment the
breaching operation. (Ref. 13,14] Ideally, for a leading battalion on the main axis of the assault some 6-8 lanes may be ordered, one for each platoon (Ref. 17,18).

Other obstacles to Soviet movement, such as rivers or blown bridges, can be overcome by utilizing the vast array of Soviet bridging equipment. Tank ditches or road craters can be countered by the bulldozer blade equipped tank.

This brief description of the Soviet counter-obstacle tactics and capabilities is as they exist today. The description is by no means complete, as that would fill several volumes. There exist many sources for information in this area. There is some problem though, in that many of the sources do not agree exactly with each other. The one area that seems to have universal agreement though, is that the capability of the Soviet forces is awesome. Perhaps this is due to the fact that the Soviet leaders recognize the necessity to continuously upgrade their forces and equipment.

The research for this chapter provides the basis for the concepts that are modelled for the Soviet forces in the STAR Engineer Effects Module.
The next chapter will discuss and describe some of the engineer effects modelling efforts that presently exist in the combat modelling community. These models have provided some useful ideas in the effort to provide an Engineer effects module for STAR.
IV. OTHER MODELLING IDEAS

This chapter will briefly discuss the relevant areas of three models currently in existence. The information available to the authors lacks computer code and is very general in nature. The purpose of this chapter is to highlight some existing modelling ideas that were of some use in the construction of the STAR Engineer Effects Module. The framework for this discussion is not to explain each model in detail, but to highlight each in the following areas:

1. Obstacle representation
2. Actions at obstacles
3. Actions out of obstacles

A. COUNTERCOM

The first model examined was COUNTERCOM, a combat simulation, developed by BDM Corporation, for evaluating the worth of military equipment and tactics in Company and Battalion level Armor/Mechanized Infantry warfare. [Ref. 20]

1. Obstacle Representation (COUNTERCOM)

COUNTERCOM models the minefield in a patterned representation. The minefield has several rows in a
minefield segment and each row is separated by an "inter-row spacing". Each row is a series of points (mines) separated by a fixed intermine spacing.

2. **Action at Minefields (COUNTERCOM)**

COUNTERCOM models mine rollers as a breaching device with an effectiveness parameter attached to count mine hits on the roller. The documentation pertaining to mine rollers does not explain the rollers representation or how its effects are modelled in the minefield. One other item of interest though, is how COUNTERCOM offsets following vehicles around dead tanks in the minefield. This is a desirable action to be modelled, however, the vehicles are allowed to pass too close to each other and thus, the tracks of the passing vehicles would actually touch the vehicle being passed.

3. **Actions Out of Obstacles (COUNTERCOM)**

COUNTERCOM models the SLUFAE (Surface Launched Unit Fuel-Air Explosive) mine clearing system as well as mine clearing line charges. These systems are Engineer unit employed and are beyond the scope of this discussion. Future efforts to install the actual Engineer units and equipment in STAR may find this pertinent to that effort. The bypass
tactic is not modelled in COUNTERCOM. In this model, Engineer units are not modelled, however, certain Engineer equipment is. Indirect fire is played against the defender but not against the attacker. The deliberate breach does not suffer the effect of indirect fire.

B. MINEFIELD AND BARRIERS COMBAT SIMULATION

The second model examined was the Minefield and Barriers Combat Simulation, developed by a British firm, SCICON. [Ref. 21]

1. Obstacle Representation (SCICON)

SCICON uses a row minefield similar to that previously mentioned for COUNTERCOM. The start point for the mines is also Monte Carlo'ed in this model. SCICON represents a scatterable minefield by assuming a totally random representation. The distance to a mine detonation for a vehicle is determined by locating the mines in the vehicle's path. The mean number of mines which fell within the vehicle's lane is found by using the density of the mines and the area of the path. This number may be used or the actual number of mines could be sampled from a binomial distribution. These mines are then placed randomly in the path of the vehicle. It is felt by the authors that the
average distance to a mine detonation could be determined from the reciprocal of the product of the minefield density and the vehicles width. This expected distance could serve as the mean of an exponential distribution draw. This would yield more realistic results and allow the model to benefit from the memoryless property of this distribution.

2. **Actions at Obstacles (SCICON)**

SCICON is reputed to play plows, rollers, and plow/roller combinations, however, from the information available it is difficult to ascertain their representation and their use within the minefields.

3. **Actions Out of Obstacles (SCICON)**

This model simulates both the breach and the "bull through" options through obstacles. It is difficult to discuss how this is accomplished due to a lack of computer code. In the documentation, however, the divert tactic and how it is modelled is discussed at great length. A route to circumvent an obstacle is planned in advance of the simulation. This route is called a branch route stream. The branch route stream option is triggered at a barrier decision point. A barrier decision point is a point on the route, a route node, that is located, such that, it precedes
a known barrier. At this decision point, units stop to determine which actions to take in order to negotiate the obstacle. This is a kind of pseudo-dynamic movement, planned for on input, but implemented by tactical outcomes. This is a useful concept and its implementation in STAR would pave the way for dynamic route selection.

C. CASTFOREM

The final model to be discussed is CASTFOREM (Combined Arms and Support Task Force Evaluation Model), developed by the U.S. Army Tradoc Systems Analysis Activity. [Ref. 22] This is a stochastic, event sequenced, opposing force simulation of ground combat. This model has an Engineer process module that plays both organic and non-organic engineering capability and virtually any engineering task for which the user has the performance data. This model will be discussed in a different format than the previous models. This is due to the fact that this model is still under construction and the information available is in outline form. CASTFOREM is projected to be able to model Engineer tasks explicitly.
1. **Methodology**
   * Task feasibility checked
   Personnel availability
   Equipment availability
   Locational resources availability
   * Task time simulation
   * Task assessment is performed
   Resource integration
   Resource disintegration

2. **Tasks Modelled**
   - Emplace minefield
   - Emplace road crater
   - Emplace AVLB
   - Breach minefield/road crater
   - Construct protected positions

   It appears on examination that this model will be quite data intensive in that it will demand detailed decision logic tables and performance data. Its general framework is complicated, yet it does not currently possess the capability to model tank ditches or bridge demolitions. No doubt this will be a capability in the future.
D. CONCLUSIONS

In order to avoid reinventing the wheel, the current state of the art in the community has been examined and briefly described herein. This preliminary research has provided several useful ideas and given the authors a perspective on the current state of the modelling of Engineer effects. The next chapter consists of the detailed description of the minefield, obstacle model and its associated counter-obstacle tactical responses.
V. THE MINEFIELD MODEL

A. INTRODUCTION

This chapter discusses the simulation of the effects of a minefield in the STAR Combat model. Implicit in this task is the construction of the minefield and the modelling of its effects on the combat process. This model allows minefields to be emplaced, cause casualties, impede the attacker, and cause the attacker to execute predetermined tactical options.

B. MINEFIELDS MODELLED

In order to model the effects of a minefield on the tactical situation on a battlefield it is first necessary to construct the model of the minefield. However, prior to the discussion of how the minefields are modelled, a brief discussion of what is used to model the minefields, the STAR field module, is in order.

The framework for obstacle simulation in STAR is the field module. Fields are represented as elliptical areas on the battlefield which influence battle actions and outcomes. In order to cause actions for entities to take place, their
relation with respect to fields must be known. Rather than have actions take place at a specific point in time, actions are triggered by certain interactions of entities with fields. These interactions are results of either field boundary crossings, entries and exits, or field internal actions. The manner in which a boundary crossing or an internal action is activated takes place in the STAR MOVE routine. In the MOVE routine, each time an entity moves, the field boundary distance (FLD.BDY.DIST), to the nearest field is decremented. When this value reaches zero, then a field boundary action takes place. Similarly, when an entity is already in a field, a field internal distance (FLD.INT.DIST), the distance to the next field internal action, is also decremented. When the FLD.INT.DIST of an entity reaches zero, an internal action is initiated. Both these distances are attributes of moving entities on the STAR battlefield. Obstacle representation and the simulation of the resulting synergistic effects became a reality through the use of the existing STAR Field Module. For a more detailed description of this Module and its function in the STAR model the reader is referred to Reference 3.
The first section of this chapter contains information pertaining to the actual modelling of a minefield in STAR. In this model, five types of minefields are simulated, they include, the anti-vehicular minefield both patterned and scatterable, the anti-personnel minefield both fragmentation and blast, and the Claymore.

1. The Anti-Vehicular Patterned Minefield

The patterned anti-vehicular minefield consists of a series of belts with mines located on the belts. These belts may be considered as lines with the mines represented as points on the lines. In this respect, the mines are considered to be dimensionless. The belts are numbered sequentially as they are encountered. The odd numbered belts have mines located at even numbered mine locations, i.e. at 2, 4, 6, ... meters. The even numbered belts have mines located at odd numbered locations, i.e. at 1, 3, 5, ... meters. Thus, each belt has a constant density of .5 mines per meter of minefield frontage. The standard planning packages for U.S. patterned minefield densities are multiples of this constant density. [Ref. 7] In addition, the use of .5 also makes the simulation of the mine's location with respect to the vehicle much easier to
accomplish. Each belt simulates three rows of a minefield. In an actual minefield, three separate rows placed in a span of approximately 16 meters forms one belt. In order to simplify the process of modelling this activity, one belt of this model actually represents the projections of the three rows onto one single line.

This basic belt construct is used in conjunction with the existing STAR field module, to model the patterned minefield. [Ref. 3] The basic idea is that the mine belts will be superimposed on the field ellipse. The density of mines (per meter of front) of the minefield is used to determine how many belts are necessary. Once the number of belts is determined, then the size of the field ellipse needed can be determined. The size of the ellipse is computed in the following manner:

The semi-major axis equals the total width of the minefield divided by 2.

The semi-minor axis is determined by the formula: number of belts plus 1 multiplied by the belt spread in meters minus 10 meters.

This total is then divided by 2.

The resulting value is the length of the semi-minor axis of the ellipse. In symbols:

$$\text{semi-minor axis} = (((\text{num belts} + 1) \times (\text{belt spread})) - 10) / 2$$
The following sketch of an elliptical minefield with the mine belts superimposed should help to clarify the construction.

![Diagram of an elliptical minefield with mine belts superimposed.]

**Figure 1: Minefield Representation**

With the previous formula in mind, an example will be worked. A minefield with a density of 1.5 mines per meter of front, a belt spread of 50 meters, and 500 meters wide is desired. Using the formulas above, the semi-major axis will first be computed.

1. The semi-major axis of the ellipse is the width of the minefield divided by two, thus, in this case it is 500/2 or 250 meters.

2. The formula for the computation of the semi-minor axis length yields the following:
   a. number of belts required equals the density desired divided by the density in each belt. For this minefield it is 1.5/1.5 = 1
   b. ((number of belts+1)*belt spread-10)/2 = ((4*50)-10)/2 = 95 meters.
Thus, for this minefield the semi-minor axis is 95 meters and the semi-major axis is 250 meters.

Finally, in modelling the patterned minefield two assumptions are made:

The minefield is homogeneous in nature.

The mines may be played in the model in various states of masking, these states range from buried and visually undetectable to being placed on the ground and therefore, subject to avoidance.

2. The Anti-vehicular Scatterable Minefield

The anti-vehicular scatterable minefield is represented as a uniform distribution of mines throughout the area an elliptical field. The basic concept of the simulation of a scatterable minefield is that an expected distance to mine encounter is determined. The technique for determining the distance to a mine encounter will be described in detail in the following section. In order to emplace a minefield the following steps are involved:

a. Select the desired minefield size by specifying the semi-major and semi-minor axis lengths.

b. The minefield density is computed by dividing the number of mines in the minefield by the area of the ellipse.

In symbols: Minefield Density = number of mines / area of ellipse

c. The expected distance to a mine encounter is equal to the reciprocal of the width of the vehicle times the density of the minefield.

In symbols:
Expected distance to encounter

\[ = \frac{1.0}{\text{vehicle width} \times \text{minefield density}} \]

The expected distance to a mine encounter is then used as the mean value for a sampling from an exponential distribution. The value that is drawn is the distance to a mine encounter for the particular vehicle concerned. Repeated draws in this manner yield the distances to each successive mine encounter and the vehicle moves through the minefield in this fashion. This minefield is also assumed to be homogeneous in nature.

3. **The Anti-personnel Blast Minefield**

The anti-personnel blast minefield is represented in a manner that is similar to the methodology utilized for the Anti-vehicular scatterable minefield. The mines are assumed to be uniformly distributed throughout the area of the elliptical field. The minefield density is computed in the same manner as the scatterable minefield discussed above. The expected distance to a mine encounter for a dismounted infantry entity is then determined by utilizing the Army Small Arms Requirements Study (ASARS) formula. A random number draw from a Uniform \((0,1)\) distribution is made and the draw is then substituted in the following formula:

\[
\text{Distance to a mine encounter} = -\frac{(2.5) \times \ln(1\text{-draw})}{\text{minefield density} \times \text{foot width}} \quad [\text{Ref. 23}] 
\]
If the entity is a vehicle, then the distance to a mine encounter is computed by drawing from an exponential distribution with a mean equal to the expected distance for a vehicle mine encounter. This expected distance is equal to three divided by the product of the minefield density and the width of the vehicle. An assumption is made that the tracks/wheels of a vehicle constitute one third of the vehicle width.

Expected distance for vehicle mine encounter
\[ = \frac{3.0}{\text{minefield density} \times \text{width of the vehicle}} \]

Thus, the entity moves through the field by distances from mine encounter to mine encounter.

4. **The Anti-personnel Fragmentation Minefield**

There are two types of anti-personnel minefields considered in this model. The types are the regular fragmentation mine and the Claymore.

a. **Regular Fragmentation Mines**

The regular anti-personnel fragmentation mine is represented as a circle with a square superimposed on it. The tripwires of the mine are represented as extensions of the diagonals of the superimposed square. (See Figure 2) The mines are placed in an orientation that favors the attacker. This is accomplished by orienting the mine so
that the attacker will always hit the tripwire at a 45 degree angle. The actual length of the tripwire exposed to the attacker is the product of the cosine of 45 degrees and the length of the tripwire. The reason for this conservative action is to allow for cases in real life when the tripwire does not deploy as is intended by design. Thus, the total tripwire length that is exposed to the attacker is really the projections of the effective lengths of each tripwire onto a line perpendicular to the direction of movement of the attacker. The length of the tripwires available to detonate the mine (effective triggering width) is equal to two times the effective length of the tripwire. (See Figure 3) The minefield density is found in exactly the same manner as for the scatterable minefield.
The expected distance to a mine encounter is computed by using the reciprocal of the product of the minefield density and the effective trigger width of one mine. This expected distance is then used as the mean for a sampling from an exponential distribution in order to determine the actual distance to the mine encounter. Vehicles are assumed to not be subject to the effects of a fragmentation mine, and thus, need not have expected distances computed; they are allowed to move freely through the minefield with no effects from the mines. Vehicles do, however, set the mines off and thus, decrease the density of the minefield. In this iteration of the module, dismounted infantryman in the area of antipersonnel fragmentation mines detonated by vehicles, are not subject to assessment as casualties from the detonations. The dismounted infantrymen are however, subject to the
effects of mines detonated by dismounted elements in their respective platoons.

b. Claymore Fragmentation Mines

The effective casualty area of a Claymore mine is a 60 degree arc with a radius of approximately 100 meters. The activation area of a Claymore is represented by an ellipse with a semi-minor axis of 14 meters and a semi-major axis of 21 meters.

![Figure 4: Claymore Mine Representation](image)

The semi-major axis is oriented parallel to the direction the attacker is moving. Once the ellipse is entered, the distance to the mine encounter is determined to be twenty percent of the length of the semi-major axis. (See Figure 4)
When the entity has travelled that distance, the Claymore detonates and casualties are assessed based on the effective casualty area for a Claymore mine. Any member of the triggering entity's platoon within the effective casualty area of the Claymore at the time of detonation may be assessed as a casualty from the blast.

This concludes the general discussion of the representation of minefields in STAR. The discussion now centers on the subject of what the physical representation yields in the STAR model.

C. THE MINEFIELD IN STAR

The model of a minefield must produce two pieces of information: the distance an entity travels prior to a mine encounter and the consequences of that mine encounter. In order to understand how this information is obtained, the following discussion explains the function of the minefield in STAR. In the MOVE routine, the model is continuously computing the distances to fields, so that appropriate actions may take place. These actions may take any of three forms, field entries, field internal actions, and field exits. During the discussion of the three types of actions, several key routines are also described. A thorough
understanding of these key routines is a necessity if one is to understand this simulation of Engineer effects on the combat process.

1. Minefield Entry Actions

In the model, as an entity crosses a field boundary a test is made to determine the type of field being entered. Once the field is determined to be a minefield, then the distance to a mine encounter is computed. This computation takes place in the routine named MINE.SCHEDULE. (See Figure 5)

a. Routine MINE.SCHEDULE

In MINE.SCHEDULE the distance to a mine encounter will be determined. First, the major category of the minefield is determined.

(1) Anti-vehicular Minefields. The second action is a determination of the system type of the entity that crossed the boundary. If the entity is a dismounted infantryman, then the minefield has no effect on the entity and he is allowed to proceed through the minefield. This is due to the assumption that a dismounted infantryman is not able to activate anti-vehicular mines. If the entity is a vehicle, then the mine type is checked. The mine type
Figure 5: Routine MINE.SCHEDULE Flowchart
dictates whether the minefield is scatterable or patterned. Once the type minefield is determined, then the distance to a mine encounter is computed.

If the type of minefield is scatterable, the computation of the distance to a mine encounter takes place as follows:

1. The minefield density is determined by dividing the total number of mines in the field by the area of the field.

2. The width of the appropriate vehicle is obtained from the vehicle dimension tables.

3. The expected distance to a mine encounter is equal to \( 1.0 / (\text{width of vehicle} \cdot \text{minefield density}) \).

4. The actual distance to the mine encounter is determined by sampling from an exponential distribution with mean equal to the expected distance computed above.

If the minefield is a patterned or hand-emplaced minefield the distance to the mine encounter is determined as follows:

1. The direction of movement is determined for sign i.e., is it positive or negative. If the direction is negative, then it is converted to the appropriate angle on a 360 degree reference system. For example, -90 degrees is equal to 270 degrees.

2. Once the direction of movement is converted, it is then translated to the coordinate system relative to the orientation of the minefield. The angle necessary for the translation is called THETA, and will be used in further computations.

3. The distance to the minebelt encounter is equal to the belt spread divided by the absolute value of the sine of THETA.
(2) **Anti-personnel Minefield.** If the minefield type is determined to be anti-personnel, then the following actions take place:

a. The minefield is checked to see if there are still any undetonated mines remaining in the field. If all the mines have been detonated, then the routine returns with no further action.

b. If the minefield is still an active casualty producing field, then a test is made to determine if the mine type is blast or fragmentation

1. If the field is a blast type minefield, then the activator is checked to determine if a dismounted infantryman. A dismounted infantryman receives a distance to a mine encounter computed by using the ASARS formula. If the activator is a vehicle, then the distance to a mine encounter is computed using the vehicle dimensions, minefield density, and a sample from an exponential distribution as previously described.

2. If the mine is a fragmentation mine, then a check is made for Claymore mines. If the mine is a Claymore, then the distance to activation is
computed. When the fragmentation mine is not a Claymore, then the activator is identified. The expected distance to a mine encounter for a vehicle is based on the reciprocal of twice the effective tripwire length plus the vehicle's width. This expected distance is used as the mean of an exponential distribution sampling. For a dismounted infantryman, the distance to a mine encounter is computed based on the tripwire length and a sampling from an exponential as discussed previously.

c. Finally, the density of the minefield is decreased for the next activator by subtracting one from the total remaining mines in the field.

Thus, the action at a minefield entry will return the distance to the next mine encounter for the scatterable anti-vehicular minefield, the distance to the next mine belt for the patterned anti-vehicular minefield, and the distance to a mine encounter for both anti-personnel minefields and the distance until detonation/activation for a Claymore mine. The entity attribute FLD.INT.DIST is now assigned the value of the distance returned from
MINE.SCHEDULE. Once the entity travels this distance, then the next action, the field internal action takes place.

2. Minefield Internal Actions

The entity has travelled the appropriate FLD.INT.DIST in the MOVE routine. An internal action for a mine encounter now occurs. During a minefield internal action, several actions may take place. In general, the actions include such events as the slowing of the speed of the vehicle, the lowering of the mine plow, the assessment of mine hits on the vehicle, and the altering of the platoon's formation. In this section of the chapter, the assessment of mine hits on a vehicle is discussed. The remaining internal actions are discussed in the detailed description of the minefield internal actions in section E.2.b. of this chapter. During an internal action Routine POP.A.MINE is called. (See Figure 6)

a. Routine POP.A.MINE

POP.A.MINE does the following:

1. Determines the number of mines hit, if any.

2. Determines the location on the vehicle of any hits, i.e., a belly or a track hit and the distance from the mine detonation if the entity is a dismounted infantryman.

3. Determines the damage from the mine encounter based on the mine lethality data.
In routine POP.A.MINE, a test is made to determine if the mine can be detected and avoided, (a scatterable or a hand-emplaced type that is not buried). [Ref. 24] If the mine can be detected, then a random number is drawn to determine if the mine is avoided. When a mine is avoided, then a new mine encounter distance is determined and the entity continues to move. The next internal action will occur at the new mine encounter distance. If, however, the mine is hit, then a check is made to determine if the entity is a vehicle or dismounted infantry.

If the field is an anti-personnel minefield and the entity is a dismounted infantryman, then the minefield is checked to see if it composed of blast or fragmentation mines.

1. If the minefield contains blast mines, then the soldier will die because he has just stepped on a mine. Routine ATTRIT is called to update the simulation with respect to the fatal disposition of the soldier that caused this action.

2. If the minefield contains fragmentation mines, then the exact location of the mine is determined. Then, the distance from the mine detonation to each member of the
activator's platoon is computed. If an entity is within the casualty radius of the mine, then routine PROB.MINE is called. Routine PROB.MINE accesses the mine lethality data for the mine detonation against personnel. This data is based on the type mine, the entity's posture at the time of detonation, (standing or prone), and the distance from the entity to the mine detonation location. The lethality data returns a probability of kill that is then Monte Carlo'ed to determine the soldier's casualty status. The result of the Monte Carlo is then sent to routine ATTRIT to update the simulation with respect to the soldier's status.

When the activating entity is a vehicle, then a check is made to determine the type of minefield entered. If the minefield is an anti-personnel minefield, then no damage is assessed against the vehicle. When the type of minefield is anti-vehicular, then a test is made to determine if the minefield is patterned or scatterable. If it is scatterable, then a random number is drawn from a Uniform (0,1) distribution to determine the mine detonation location on the vehicle. An assumption is made that one-third of the vehicle bottom area is track and two-thirds is belly. The random number is then Monte Carlo'ed against
this standard to determine the location of the mine hit. Once the hit location is determined, routine PROB.MINE is called. PROB.MINE accesses the vehicle lethality data for mine hits based on the mine type, vehicle type, and the hit location on the vehicle. PROB.MINE passes the appropriate lethality data to routine ATTRIT, the damage to the vehicle is assessed, and the simulation is updated.

If the mine type/minefield is patterned anti-vehicular, the following computations are made:

1. The number of the next belt to be hit is determined.
2. The distance to the outer boundary is computed.

The outer boundary is the tangent line on the ellipse that is perpendicular to the semi-major axis. This distance is computed in several steps. The first step is to convert the X coordinate of the location of the vehicle to an X prime location in a coordinate system that uses the major axis of the field as the I axis and the minor axis of the field as the Y axis. X prime is obtained by the translation formula:

\[ X' = X \cos(\text{rotation angle}) + Y \sin(\text{rotation angle}) \]

The distance to the outer boundary (DTOB) is now calculated as the absolute value of the difference between the X coordinate value of the center of the field and the X prime
value described above. This number is then subtracted from the length of the semi-major axis of the field. This formula is applicable to both sides of the minor axis of the field. It is this distance that is the key to superimposing the mine belts on the elliptical field, and thus, the locations of the mines relative to the vehicle's position. The vehicle width (VW) is also adjusted to account for the angle of vehicular movement. Next, the integer mine location number just to the left of the vehicle (NUM.LEFT.SIDE) is determined. The routine then determines if the belt is even or odd, and consequently, if the mines are located at even or odd integer locations. The NUM.LEFT.SIDE is then adjusted to the appropriate odd or even integer further to the left of the vehicle. A loop search is then conducted to determine where the mines are located in relation to the vehicle; track, belly, or miss. Again, the assumption is made that the track takes up one third of the vehicle width. The detonation locations are found and routines PROB.MINE and ATTRIT are called to assess the damage. The routine PROB.MINE will now be described. (See Figure 7)
Figure 7: Routine PROB.MINE Flowchart
b. Routine PROB.MINE

The purpose of this routine is to access the appropriate mine lethality data and assess the damage to the entity.

For dismounted infantry the lethality data is stored in a three dimensional array called GRUNTLETH. The

Table 6: GRUNTLETH

<table>
<thead>
<tr>
<th>Mine type</th>
<th>Posture of soldier</th>
<th>Distance from mine detonation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M16</td>
<td>1 soldier standing, distance in meters</td>
</tr>
<tr>
<td>2</td>
<td>ADAM</td>
<td>1 soldier standing, distance in meters</td>
</tr>
<tr>
<td>3</td>
<td>E74</td>
<td>2 soldier prone, by 10 meter increments</td>
</tr>
<tr>
<td>4</td>
<td>Claymore</td>
<td>except Claymore data is</td>
</tr>
</tbody>
</table>

dimensions of GRUNTLETH are four by two by five.

The value that is returned from GRUNTLETH is a probability of kill.

If the mine is anti-personnel, then a check is made to determine the actual type of mines in the field. When the mine is a blast mine, then the soldier becomes a casualty. If a soldier activates a fragmentation mine then any member of his platoon may be affected. The posture of each soldier in the platoon, his distance from the mine detonation, and the type mine are used to access GRUNTLETH in order to determine a probability of kill. For each mine
detonation, a probability of kill for each live platoon member is determined. This probability is Monte Carlo’ed by drawing from a Uniform (0,1) distribution to determine if any soldiers are casualties as a result of the blast.

The mine lethality data for vehicles is stored in the array MINELETH. MINELETH is a four dimensional array that is five by eight by two by four. The data stored in MINELETH is Ballistics Research Laboratory (BRL) lethality data. It is stored as follows:

Table 7: MINELETH

<table>
<thead>
<tr>
<th>Mine Type</th>
<th>Vehicle Type</th>
<th>Detonation Location</th>
<th>Damage Probabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 #70</td>
<td>Tank</td>
<td>1 Track</td>
<td>Mobility</td>
</tr>
<tr>
<td>2 #56</td>
<td>APC</td>
<td>OF</td>
<td>Firepower</td>
</tr>
<tr>
<td>3 #21</td>
<td>Command</td>
<td>2 Belly</td>
<td>Mob/Fire</td>
</tr>
<tr>
<td>4 #15</td>
<td>BRDH</td>
<td>Hit</td>
<td>Catastrophic</td>
</tr>
<tr>
<td>5 #19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 #22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 #11</td>
<td>SP How</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 SPAT</td>
<td></td>
<td>125</td>
<td></td>
</tr>
</tbody>
</table>

Routine PROB.MINE first tests to determine the mine type. If the mine is anti-tank, then the system type of the activating entity is determined. With the type anti-tank mine, vehicle type, and the detonation location on the vehicle, MINELETH is accessed and the appropriate damage probability values are returned to routine POP.A.MINE.
At this point, the program would return from routine POP.A.MINE and call routine MINE.SCHEDULE. A new distance to a mine encounter is determined and the cycle continues until the entity becomes a casualty or exits the minefield.

3. Field Exit Actions

When an entity exits a minefield its speed is upgraded to the maximum allowed by the terrain. If a lane has been cleared through the minefield, then this information is stored for future use by follow-on elements.

The previous discussion has described how the minefield is represented in the STAR model. In the next section, the discussion will focus on the synergistic effects modelled.

D. SYNERGISTIC EFFECTS

In this section, the tactics and synergistic effects of minefields are discussed.

1. Tactical Alternatives for the Commander

A tactical commander encountering a minefield has basically three alternatives to choose from:

a. He may choose to conduct a hasty breach if the unit possesses any live plows or rollers.

b. He may bypass the minefield if the tactical situation permits.
c. He may, as a last resort, order a "bull through" operation, just driving the vehicles through the minefield hoping to clear a lane and taking casualties.

The choice that the commander makes is based on whether or not he possesses knowledge of the minefield. The manner in which the knowledge is gained is irrelevant. What is relevant though, is when the knowledge is gained. If the commander gains the knowledge prior to entering the minefield, then there is more flexibility in the selection of tactics that are most beneficial for the mission of the unit. If, on the other hand, he gains his knowledge the hard way, i.e. by detonating a mine, then the commander is forced to react in a different fashion. In the design of this model, some judgmental decisions based on knowledge and experience have been made. As a result, certain tactical alternatives were chosen for simulation in this Engineer Effects Module. A recapitulation of the tactical alternatives in this module are included in Table 8. The expected tactics of both the US and the Soviet force commanders are the basis for this model.

2. Minefield Synergistic Actions

The general minefield effects simulated will now be described. As mentioned above, the knowledge of the
Table 8: Tactical Alternatives at an Obstacle Encounter

1. Conduct a breaching operation through the obstacle whenever the platoon possesses the proper breaching equipment.
2. "Bull-through" the obstacle if specified by the user or input.
3. If the "bull through" is not specified, then the entity will:
   a. Move to a cleared lane through the obstacle, if one exists.
   b. Bypass the obstacle if a bypass exists and no clear lane exists through the obstacle.

Minefield and the mission of the force are the driving factors in the selection of the counter-obstacle tactics. In this section of the chapter, the breach and "bull through" actions are discussed. The bypass action will be discussed in Section 3. of this chapter. It is for this reason that the discussion will begin with the knowledge attribute of the field. The knowledge attribute of the field may have any of four values, 0, 1, 2, or 3.

a. 0 or 1 means that there is no prior knowledge of the existence or location of the minefield.

b. 2 means that the minefield's location, but not its extent, is known and that partial lanes through the minefield may exist.

c. 3 means that the minefield, and any clear lanes through it are known.

(See Figure 8) As an entity crosses the field boundary into a minefield, the knowledge level on the field is checked and the appropriate action is taken.
Figure 8: General Minefield Boundary Actions Flowchart
a. Actions with No Prior Knowledge

If the knowledge level is 0 or 1, then routine MINE.SCHEDULE is called to compute a distance to a mine encounter. If the minefield type is patterned or hand-emplaced, then once this distance is traversed by the entity, a detonation occurs and the knowledge level is changed to two. If, however, the mines are visually detectable, then the possibility of avoiding the mine exists. If the distance to the mine encounter is greater than the minefield visual detection distance, then the minefield is allowed to be detected without a detonation occurring. The minefield visual detection distance is designated by the user. Two studies on the subject, the TEMAWS study and a study conducted at Ft Drum N.Y. [Ref. 25,26], may be helpful in the determination of this value. The distance for the entity to travel in the minefield is set to this visual detection distance, and the knowledge level on the minefield is set to two. If on the other hand, the encounter distance is less than or equal to the visual detection distance, then routine POP.A.MINE is called, the detonation occurs, and the knowledge level on the minefield is set to two.
The knowledge level is two, and if the entity is a vehicle the next check is to determine if there are any alive plows remaining in the platoon. If no plows are alive, then the platoon turns around and leaves the minefield on the same path it entered. The next action for this exiting platoon is a decision ellipse entry action. This action is discussed in Section E.1. of this chapter. If the platoon possesses a plow that is alive, then the platoon stops for a plow activation time and changes to a plow formation (the plow leading with all other vehicles following in column directly behind the plow). The platoon also has its speed degraded by a plowing speed factor.

In using a mine plow to clear a lane through a minefield several assumptions are made about plows.

1. Plows clear all mines from the path of the tracks of the vehicle.
2. The plow suffers no degradation due to mines hitting the skids.
3. Following tanks are able to follow directly in the tracks cleared by the plow as a result of the berms created.

Once plowing begins, routine MINE.SCHEDULE is called to establish a new mine encounter distance. If the plow vehicle is still alive when this point is reached, then routine MINE.SCHEDULE is called again to compute a new mine
encounter distance. This process continues until the minefield boundary is crossed or the plow vehicle is killed by direct fire. If the plow is killed during the travel to the next mine encounter, then there are two options for the platoon, the dead plow may be pushed by the following tank or the live vehicles may offset around the dead plow and attempt to make a lane through the field without it. When the plow vehicle is dead, then a draw from a Uniform(0,1) distribution is made. The random number is Monte Carloed against a user specified probability of pushing, to determine which option, pushing or offsetting, is selected.

If the offset option is selected, then routine POP.A.MINE is called to assess mine damage to the lead tank at the location of the mine detonation. The speed of the platoon is upgraded to the maximum allowed by the terrain and routine MINE.SCHEDULE is called to determine the distance to the next mine encounter. At the mine encounter, routine POP.A.MINE is called and the damage is assessed to the lead tank. This process continues until either the platoon is depleted or the minefield is exited. If the push option is selected, the platoon’s speed is degraded further from the current plowing speed and the platoon continues to
move through the minefield just as though the plow were alive. Several assumptions are made with respect to the pushing option.

1. Another tank can push a plow tank provided that the tracks are still on it. A plow tank killed by direct fire is assumed to still have its tracks mounted.

2. A pushing tank gains a slight advantage against direct fire weapons due to the fact that it is shielded somewhat, by the tank it is pushing.

3. A platoon can only elect to push the plow once. If the pushing vehicle is killed the option to push it and the plow does not exist.

4. The pushing tank is able to follow directly in the cleared track path of the plow.

Once the pushing has begun, routine MINE_SCHEDULE is called to compute the distance to the next mine encounter. At the encounter, the push tank is checked to see if it is alive. If it is still alive, then the process continues until either the platoon exits from the field or the pusher dies. If the pusher is dead, then the offset sequence begins and continues as previously described.

The next path to be described is the case when the knowledge level is two. Again, the reference will be to Figure 8.
b. Actions with Knowledge Prior to Entry

When the knowledge level is two, upon entry into a minefield, an entity need not detonate a mine to find out that a minefield exists. The platoon is already aware that a minefield exists and first checks for an alive plow in the platoon. After this, the path is the same as described in the previous section from the point where the knowledge level was changed to two.

c. Actions When a Lane is Known to Exist

The final case to be discussed for the minefield actions is the case when the knowledge level is three. This means that there is a clear lane somewhere through the minefield. This is the final case to be described from Figure 8.

A check is made to see if the route the platoon is travelling is clear of mines. If this is not the case, then the same path as a knowledge level of two is followed. The unit will attempt to gain a breach. The reason for this is that the commander takes actions to secure additional lanes through the minefield. The desire of threat unit commanders to secure at least one lane per platoon is discussed in Chapter 3 of this thesis. If, however, the
platoon is travelling on a cleared route, then the platoon is assembled into a column formation and the speed is degraded for lane travel safety. The formation and the lane speed factor are both input parameters. In this case, two additional assumptions are made:

The follow-on platoons will be able to follow directly in the cleared, marked lanes.

If a tank is killed by direct fire while travelling on the cleared lane, the following vehicles will be able to push the dead tank out of the way and continue on the lane.

Each of the above knowledge cases eventually yields the same result, a minefield exit. Basically, there are two actions that occur at a minefield exit. If the vehicle is exiting the minefield ellipse as the result of clearing a lane, then the knowledge level of the minefield is changed to three and the platoon conditions are reset for future minefield encounters. The knowledge attribute allows follow-on units to know that this particular route through the minefield is clear. If the vehicle is not exiting as the result of clearing a lane, then it is moving in a reverse direction back along its route towards a field ellipse called a decision ellipse. The decision ellipse and its function in the module is discussed in detail in the next section of this chapter, Section D. 3. In this case,
the speed and formation are upgraded to the original formation and speed but a cleared lane is not recorded.

3. The Decision Ellipse

In the previous section, reference was made to a new type of field, the decision ellipse. This creation is the key to the capability to insert pseudo-dynamic movement into the STAB combat modelling process. The decision ellipse allows the elements in the simulation to make route changes based upon the tactical situation and the status of the platoon's organic breaching equipment. The movement has been labeled as pseudo-dynamic because the alternative route options must be planned in advance as input, however the actual selection from the alternatives is dictated by the battlefield situation.

The decision ellipse is located on a route such that it will be encountered prior to an engineer obstacle. The decision ellipse is to be located such that it affords the attacker maximum cover and concealment from enemy direct fire. At the decision ellipse, the first decision concerning the minefield is made.

The general actions that take place at a decision ellipse will now be discussed. (See Figure 9)
Figure 9: General Decision Ellipse Flowchart
The decision ellipse is linked to the obstacle ellipse that is associated with it. This is accomplished by the use of an integer obstacle ellipse name. As a vehicle moves into the decision ellipse, the knowledge level of the obstacle is ascertained by linking to the appropriate obstacle ellipse. If the obstacle ellipse knowledge level is 0 or 1, then the vehicle passes through the decision ellipse with no further action. If the knowledge level is two, then the platoon is checked for the presence of a plow. If the plow is alive, then the platoon continues to move with no further action in the decision ellipse. If the plow is not alive, then the platoon is given its bypass route and next movement control point (MCP) and moves toward the new route. If the knowledge level is three, the platoon is checked for the presence of an alive plow. If the plow is alive, the platoon will continue to move on its route through the decision ellipse with no further action until the minefield boundary is crossed. If the plow is not alive, then the platoon will obtain the route and MCP of the cleared lane and move to it.

In order to better illustrate the concept of the decision ellipse the reader is referred to Figure 10 for the following discussion of some situations.
Figure 10: Decision Ellipse Example
Each route has a platoon travelling on it. The decision ellipses were set up in the following manner:

<table>
<thead>
<tr>
<th>Decision Ellipse</th>
<th>Bypass Route</th>
<th>Bypass MCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

**Situation 1:**

RT 3 is a cleared route through the minefield. The platoon travelling on RT 4 has reached the decision ellipse and does not have a plow vehicle alive. The knowledge level of the obstacle is 3, thus, there is a cleared lane through the minefield. The decision ellipse tells the platoon to go to the cleared lane located at RT 3 and MCP 2.

**Situation 2:**

The platoon on RT 3 reaches decision ellipse B and their plow is dead. The knowledge level on the obstacle is 2 because other platoons have entered the minefield and are attempting to clear lanes through it. The platoon on RT 3 does not have a clear lane or a plow, therefore it will go to the bypass, RT 2 and MCP 2. As the platoon reaches decision ellipse A on its bypass route it will again check to see if a lane has been completed through the minefield.
If there still is no lane through the minefield, the platoon will bypass to RT 1 and MCP 2. These bypass routes and MCPs are kept as attributes of the entered decision ellipse. If however, the knowledge level is 3, because a lane had been completed while the platoon was moving from B to A, then the platoon will move to the route that contains the cleared lane. The location of the information about the cleared lanes is carried in a three-dimensional array called TRAFFIC.CONTROL. The following information is stored for each route for each obstacle:

1. Route number of the route being travelled.
2. Decision ellipse MCP on that route.
3. Cumulative distance clear in the obstacle on the route.
4. The most recent distance to a mine encounter for the lead vehicle of the lead platoon in the minefield.
5. The number of mine encounters scheduled thus far for the elements travelling on this route through the minefield.

For a more detailed discussion of the TRAFFIC.CONTROL array, see Appendix A.

In this section, the discussion has been of a general nature with regard to what is modelled. This is a good foundation and a necessary prerequisite for an understanding of the next section, which consists of a
detailed description of the actions that take place at the various locations pertaining to fields.

E. DETAILED FLOWCHART APPROACH

In this section the discussion is divided into three categories, actions at field entry, field internal actions and actions at field exits. Section C of this chapter, gives the reader an overview of the synergistic effects captured in this model. This section discusses in further detail how the desired synergistic events are captured in this model of the combat process. The reader is advised that routines POP.A.MINE and MINE.SCHEDULE play an important role in this discussion and thus, should be understood well, prior to continuing. This model is constructed in a fashion to account for many simultaneous actions occurring on the battlefield. For example, several platoons may be moving through the same minefield either on different or common routes or combinations thereof, concurrently. The ability to handle many simultaneous actions necessitates the use of many flags and tests in order for the model to discern which entities are involved in the actions. It is for this reason that the reader will notice a great number of tests and different situations covered in the following discussions. The decision ellipse will be discussed first. (See Figure 11)
Figure 11: Minefield Decision Ellipse Flowchart
The decision ellipse on each route carries the following additional attributes as information:

Table 9: Decision Ellipse Attributes

<table>
<thead>
<tr>
<th>P1.FLD</th>
<th>The bypass route for this route.</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2.FLD</td>
<td>The bypass MCP on the bypass route.</td>
</tr>
<tr>
<td>P3.FLD</td>
<td>The field number of the obstacle associated with this decision ellipse.</td>
</tr>
<tr>
<td>P4.FLD</td>
<td>The number of the route this decision ellipse is located on.</td>
</tr>
<tr>
<td>P5.FLD</td>
<td>The MCP inside the decision ellipse.</td>
</tr>
</tbody>
</table>

When an entity enters a decision ellipse, the decision ellipse attempts to link to the minefield ellipse in order to obtain the knowledge level of the minefield. The link is possible because the decision ellipse is given the integer number name of the minefield it is associated with as an attribute. If the decision ellipse is able to link to the minefield, then the route through the decision ellipse also goes through the minefield. In order to ensure that the next action is a decision ellipse exit action, the internal distance for the vehicle is set to plus infinity. If the link is not possible, then the decision ellipse is on a route that is a bypass around the minefield. If the entity is on a bypass route, then the entity must utilize the formation on the route. However, if the entity is
utilizing lateral movement in order to circumvent the obstacle, then the route formations are not used. This instance requires a more detailed examination.

In order to introduce lateral movement between decision ellipses on different routes, pseudo-dynamic movement, the formation dictated by the route had to be overridden. [Ref. 27] When using route formations, movement is conducted by the projection of the vehicles location onto the route with respect to the movement control points (MCP) on that route. The vehicle is considered to be through an MCP as soon as its projection on the route is. When moving laterally, this does not suffice. For lateral movement, the vehicle itself and not its projection must pass through the MCP. This action is accomplished by setting an existing vehicle attribute, FORMACODE, to zero. Setting FORMACODE to zero overrides the route formation and causes the entity to proceed directly to an MCP. This insures that the entity gets to the MCP, not just its projection.

At a decision ellipse, several additional actions take place. First a check is made to ensure that the entity is moving in the direction of the attack, rather than coming back from the obstacle. If the latter is true, the entity...
movement of the entity is halted and pseudo-dynamic route selection, either to a cleared lane or a bypass route, occurs. A decision ellipse may be entered in any of three different ways.

1. Moving in the direction of the attack.

2. Returning from the obstacle, (moving opposite to the direction of attack).

3. Moving from the flank, (lateral movement from another decision ellipse).

Each direction of entry into the decision ellipse has certain tactical alternatives associated with it. The following table associates the tactical alternatives of the module listed in table 8, with the appropriate direction of entry into a decision ellipse.

<table>
<thead>
<tr>
<th>Direction of Movement</th>
<th>Alternatives from Table 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the direction of the attack</td>
<td>1, 2, and 3</td>
</tr>
<tr>
<td>Returning from obstacle</td>
<td>3</td>
</tr>
<tr>
<td>From the flank of ellipse</td>
<td>3</td>
</tr>
</tbody>
</table>

The remaining internal actions are identical to those discussed in the general case in section D.3., above. At an exit from a decision ellipse, no further actions take place.
2. **The Minefield Ellipse**

The minefield ellipse actions are probably the most complicated of the actions to be discussed. As in the case of the decision ellipse, the minefield ellipse possesses attributes which are used to control the actions in the minefield. These attributes are listed below.

**Table 11: Minefield Ellipse Attributes**

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Scatterable minefield</th>
<th>Pattern minefield</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1.FLD</td>
<td>number of mines in field</td>
<td>state of minefield activation</td>
</tr>
<tr>
<td>P2.FLD</td>
<td>trip wire length</td>
<td>belt separation</td>
</tr>
<tr>
<td>P3.FLD</td>
<td>mine type</td>
<td>mine type</td>
</tr>
<tr>
<td>P4.FLD</td>
<td>probability of detection and avoidance</td>
<td></td>
</tr>
<tr>
<td>P5.FLD</td>
<td>probability of pushing a dead plow tank</td>
<td></td>
</tr>
<tr>
<td>P6.FLD</td>
<td>knowledge level (0,1,2, or 3)</td>
<td></td>
</tr>
<tr>
<td>P7.FLD</td>
<td>cleared lane formation number</td>
<td></td>
</tr>
<tr>
<td>P8.FLD</td>
<td>cleared lane speed factor</td>
<td></td>
</tr>
<tr>
<td>P9.FLD</td>
<td>plow activation delay time</td>
<td></td>
</tr>
<tr>
<td>P10.FLD</td>
<td>plowing speed degradation factor</td>
<td></td>
</tr>
<tr>
<td>P11.FLD</td>
<td>pushing speed degradation factor</td>
<td></td>
</tr>
<tr>
<td>P12.FLD</td>
<td>minefield visual detection distance</td>
<td></td>
</tr>
</tbody>
</table>

a. **Detailed Minefield Entry Actions**

Upon entry into a minefield the first action that takes place is to determine if the minefield is activated. (See Figure 12) If the minefield is not activated, the entities move through the field as if it did not exist. The entity entering the minefield is then checked to determine if it is a dismounted infantryman or a vehicle. If the entry is made by a dismounted infantryman,
Figure 12: Minefield Entry Action Flowchart
then routine MINE.SCHEDULE is called and the actions previously discussed take place. The reason for this is that dismounted infantrymen do not activate anti-vehicular mines. Immediate action drills by infantrymen are not modelled. When a vehicle enters a minefield, a check is made to ascertain if there are bypass routes for this minefield and route. If a bypass route does not exist, then the knowledge attribute is set to one.

(1) **No Prior Knowledge Exists.** When the knowledge level is zero, the following sequence of events takes place. Routine MINE.SCHEDULE is called and a distance to a mine encounter is determined. This becomes the field internal distance (FLD.INT.DIST). A test is then made to determine if the minefield can be visually detected. When the minefield can be visually detected, a test is made to determine if the minefield will be detected prior to travelling the distance to the mine encounter. [Ref. 25,26] When the visual detection distance is greater than zero, it is compared to the computed distance to the mine encounter (FLD.INT.DIST). The shortest distance of the two then becomes the distance to the next internal action. If the visual detection distance is the smaller distance, the
knowledge level is set to one without a mine detonation occurring. If the mine detonation distance is the smaller distance, there will be a mine detonation at the internal action and the knowledge level will be set to two. Whichever distance is selected, the route's cumulative distance clear and platoon movement number get updated in the TRAFFIC.CONTROL matrix. Logic control is then returned to the MOVE routine.

When the knowledge level is one on entry, the TRAFFIC.CONTROL matrix is checked to see if the cumulative distance clear on the route being traversed is greater than zero. If the cumulative distance clear is not greater than zero, the no knowledge path is followed. The path for the no knowledge level is described above. If the value is greater than zero, the vehicle entering the field will have its movement number changed to the platoon movement number in the TRAFFIC.CONTROL matrix and the FLD.INT.DIST is set to the cumulative distance clear on this route. Follow-on vehicles travel safely on a path as far as other vehicles have already travelled. In order to gain this and unit movement, the TRAFFIC.CONTROL matrix was created. A complete discussion of TRAFFIC.CONTROL is contained in Appendix A.
(2) Knowledge of the Minefield Exists. When the knowledge level is two, a check is made to determine if the platoon has an alive plow. When there is no plow alive, the entire platoon is turned around, to return to the decision ellipse, by switching the platoon's starting and ending areas. In addition, the FLD.INT.DIST is set to plus infinity to ensure that the field boundary will be crossed. If an alive plow exists in the platoon, then TRAFFIC.CONTROL is checked to determine if this route through the minefield has a cumulative distance cleared. If the cumulative distance clear on the route is greater than zero, then some of the route through the minefield is clear. The FLD.INT.DIST is then set to the cumulative distance cleared, the entity moves to the cleared distance, and then control is returned to the MOVE return. If the cumulative distance clear is zero, then the platoon is stopped for the time it takes to activate the plow and attain a plowing formation (the plow in front with all vehicles following directly in the plow's path). The speed of the platoon is degraded by a factor to account for a slower plowing speed and the PLATOON.COND is set to two (plowing). The plow speed degradation factor is a user specified value. Routine
MINE.SCHEDULE is called to get the distance to the next mine encounter; TRAFFIC.CONTROL is updated; and logic control is returned to the MOVE routine.

(3) **A Lane Exists in the Minefield.** The final path in the minefield entry case is the when the knowledge level is three. This means that at least one clear lane exists through the minefield. The first action to take place is to determine the cumulative distance clear on the route being travelled from the TRAFFIC.CONTROL matrix. If the value is plus infinity, the cleared lane is on the route the platoon is currently traversing. The FLD.INT.DIST is set to plus infinity, the formation is changed to a column, and the speed that the platoon utilizes on the cleared lane is degraded by a speed factor that the user inputs during the initial battle set up. If the cumulative distance clear is not equal to plus infinity, then the path for a knowledge level of two is followed.

This concludes the detailed description of the minefield entry actions. The next section is a description of the minefield internal actions. (See Figure 13)
b. Detailed Minefield Internal Actions

In the MOVE routine, the FLD.INT.DIST values are continuously checked for each moving entity. Once the FIELD.INT.DIST is zero, then an internal action is triggered. The first test is to determine if the entity triggering the internal action is a dismounted infantryman or a vehicle. If the entity is a dismounted infantryman, then routine POP.A.MINE is called and the damage status of the entity is determined. When the entity is still alive, routine MINE.SCHEDULE is called, a new internal distance is computed, and control returns to the MOVE routine. When the entity is not alive, control is returned to the MOVE routine and no other actions take place. If, on the other hand, the activating entity is a vehicle, then the knowledge level of the minefield is checked.

(1) No Knowledge Exists (0 or 1). If the knowledge level is zero, then the first action is to call Routine POP.A.MINE. In this routine, the damage caused by the mine detonation is assessed. The knowledge level on the field is changed to two. If the knowledge level is one, a visually detected minefield, then routine POP.A.MINE need not be called in order to set the knowledge level to two.
Figure 13: Ninefield Internal Actions Flowchart
The next check is to ascertain if the unit is utilizing the "bull through" tactic. If the action is not a "bull through", then the next check is to determine if a plow is alive in the platoon. If a plow is alive, it is activated, the plow formation is set, and control is returned to the MOVE routine. In the case where no plow remains alive in the platoon, then the platoon is turned around and sent back to the decision ellipse. If the unit is utilizing a "bull through" tactic, the platoon follows the path for a knowledge level of two. This ends the description for the instance of knowledge levels of zero or one.

(2) Knowledge Exists(2 or 3). When knowledge exists about the minefield, the first check is to determine if a clear lane exists on the present route. If a lane exists, then the FLD.INT.DIST is set to plus infinity, the formation is changed to a column, the speed is degraded, and control is returned to the MOVE routine. If the entire route is not clear, then the distance clear to travel (FLD.INT.DIST) is updated to the latest cumulative distance clear on the route listed in the TRAFFIC.CONTROL matrix. Next, a test is made to determine if a plow is still alive in the platoon. If there is a plow still alive, then the
PLATOON.COND is checked to see if the plow is already plowing. If it is not already plowing, then the actions to get the platoon into a plowing configuration are called and a hasty breach is begun. If the plow is already plowing, then routine MINE.SCHEDULE is called yielding a new distance to a mine encounter and the TRAFFIC.CONTROL matrix is updated. Control is now returned to the MOVE routine. If there is no plow still alive in the platoon, then the platoon plowing condition serves as the next branch point for our actions. To review the situation, the platoon is in a minefield, has knowledge of the minefield, and there are no plows alive in the platoon. If the PLATOON.COND is equal to one, the platoon had a plow upon entry into the minefield, but had never employed it. At this point, the platoon is sent back to the decision ellipse and control returns to the MOVE routine. If the PLATOON.COND is two, then the platoon had been plowing already. This means that the platoon has two courses of action available to it, it can elect to push the dead plow, or offset around it with the remaining live vehicles. The choice between the two options is a probability that is specified by the user. A random number is drawn from a Uniform (0,1) distribution and
Monte Carlo'ed to determine which choice will be taken. If the pushing option is selected, then the actions that take place are:

1. The platoon speed is degraded to the speed of a tank being pushed.
2. The PLOWCOND of the push tank is set to four in order to distinguish the push tank from the other vehicles of the platoon. The PLATOONCOND is set to four, indicating that the platoon is utilizing the pushing option.
3. The defilade number (DEFMUN) of the pushing tank is changed. This change affords the pusher partial cover from direct fire.
4. Routine MINE.SCHEDULE is called and the TRAFFIC.CONTROL matrix is updated.

The selection of the offset option would trigger the following actions:

1. The speed of the platoon would be upgraded back to the maximum speed allowed by the terrain.
2. The activating vehicle has its PLOWCOND set to three, indicating that it is the lead vehicle of the platoon. The PLATOONCOND is also set to three indicating that the platoon is in the offset sequence.
3. Routine POP.A.MINE is called and damage to the lead vehicle is assessed.

4. If the lead vehicle is still alive, then routine MINE.SCHEDULE is called, TRAFFIC.CONTROL is updated, and control is returned to routine MOVE.

5. The formation of the platoon is changed to an offset formation and the mine type is checked. The mine type is checked because in a pattern minefield, offsetting vehicles are allowed to hit the same belt that killed the lead vehicle. If the type is scatterable, it is an expected distance that is computed for the offsetting tank to move prior to a mine encounter.

The final check at this branch is for a PLATOON_COND of three or four. This means that the platoon had already been pushing or offsetting. The status of the lead tank or the push tank is checked. If this tank is still alive, then routine MINE.SCHEDULE is called to determine the distance to the next mine encounter and the TRAFFIC.CONTROL matrix is updated. If this vehicle is dead, then the offset option is chosen. It is felt that the task of pushing one dead tank that has a plow on it will be very difficult; the task of pushing two dead tanks, impossible.
Thus, from this point on, the remaining vehicles in the platoon will have to offset around any dead vehicles. Their actions will follow the offset case previously described.

This completes the detailed description of the minefield internal actions. The final point of discussion in this chapter is actions to be taken at an exit from a minefield.

c. Detailed Minefield Exit Actions

The actions that take place upon exit from a minefield are relatively easy to understand when one compares them to the actions discussed already. The reference for this discussion is Figure 14.

The first action to take place on exit is to upgrade the vehicle speeds to the maximum allowed by the terrain. The second is to determine if the entity is exiting on the way back to a decision ellipse associated with this minefield, or moving in the direction of the attack. If the movement is in the direction of the attack, then a lane has been cleared for other units to follow. The following values are then set:

1. The knowledge level on the field is set to three. (a lane exists)

2. In the TRAFFIC.CONTROL matrix the cumulative distance cleared for the route is set to plus infinity.
Figure 14: Minefield Exit Actions Flowchart
3. The platoon has its formation reset to the formation directed by this route.

4. The vehicle attributes pertaining to minefield actions are reset so that future minefield encounters will not be affected by this breach. Control is returned to the MOVE routine.

If the entity is moving back to a decision ellipse on exit, then the platoon has its formation reset to the one used on entry and continues its movement.

F. CONCLUSION OF THE MINEFIELD MODEL

This concludes the discussion of the model itself and how the actions take place. The serious user of this model should be able to take this description, the STAR running instructions, and the SIMSCRIPT II.5 code included in the appendices and have minefields and their synergistic effects in a STAR Combat simulation run. In the next chapter, engineer gap obstacles, such as, the tank ditch, road crater, and blown bridge (short-wet gap), are discussed. The discussion of the modelling of these obstacles will generally follow the same format as the description in this chapter.
VI. THE ENGINEER GAP OBSTACLES

A. INTRODUCTION

This chapter discusses the simulation of the Engineer gap obstacles in the STAR Combat Model. As in the case of the minefield, first the obstacles must be modelled and then the effects of the obstacles on the combat process can be simulated. The gap obstacle portion of the Engineer Effects Module allows obstacles to be emplaced, impede the attacker and cause the attacker to execute predetermined tactical options. Several of the new methods and terminologies introduced in the minefield chapter are also used with the gap obstacles. Consequently, this chapter requires a somewhat less detailed explanation than previous discussions.

B. GAP OBSTACLES MODELLED

In order to model the effects of the gap-type obstacles on the combat process, it was first necessary to install the gap obstacles in the STAR model. The Engineer gap obstacles that are modelled and described in this chapter include, the tank ditch, road crater, and blown bridge (short-wet gap).
1. **The Tank Ditch**

   The tank ditch is a linear obstacle that is modeled as a path 3.3 meters wide, corresponding to the major axis of a field ellipse. The size of the ellipse is determined by two factors, the actual length of the tank ditch and its visual recognition distance. The length of the semi-major axis of the ellipse is equal to half the length of the tank ditch, and the semi-minor axis is the visual recognition distance. (See figure 15)

   ![Figure 15: Tank ditch representation](image)

2. **The Road Crater**

   The road crater is a point type obstacle, represented as the center of a field ellipse, a circle
located on a road or route. The radius of the circle is the visual recognition distance. These craters (circles) are placed on the suspected routes of the attacking force and are designed to delay and disrupt the channelized movement of this force. (See figure 16)

![Figure 16: Road Crater/Short-Wet Gap Representation](image description)

3. **The Blown Bridge (short-wet gap)**

   The short wet gap is represented in a manner that is very similar to the road crater. The field ellipse representing the short-wet gap is constructed so that the route or road is covered by the circle at the desired point. (See figure 16) Vehicles cannot proceed past this point without bridging assets. In this model, swimming or snorkeling of vehicles across the wet gap is not simulated.
C. THE GAP OBSTACLE IN STAR

The simulation of the gap obstacle in STAR produces certain synergistic effects, one such effect is the selection of tactics to overcome/neutralize enemy obstacles. In Chapter five, three minefield tactical alternatives for the commander were specified. These same three alternatives are appropriate for gap type obstacles and are also modelled. For the reader's convenience, the alternatives are listed below.

- Breach the obstacle
- Bypass the obstacle
- "Bull Through" the obstacle

As in the minefield model, the control mechanism for the implementation of the above three alternatives is the decision ellipse. The status of a unit's organic breaching equipment, however, still serves as the major factor in the selection of the alternative tactical responses. The types of breaching equipment are different but the concept remains the same. In addition, some breaching equipment is only effective against specific gap obstacles. For example, a dozer blade equipped tank cannot breach the short-wet gap, however, it can be used to fill the tank ditch or road.
crater with earth to allow passage of vehicular traffic. The Armored Vehicle Launched Bridge (AVLB), on the other hand, is effective against all the gap obstacles, since it is emplaced across the gaps created by the obstacles. Finally, the self-breach is akin to the "bull through" instance from the minefield model. This means that the vehicle can cross some obstacles by itself. This is accomplished in a rather crude fashion; the tracked vehicle oscillates back and forth in the ditch or crater, trying to cave the sides of the obstacle walls in to allow for an easier exit of the vehicle. This exposes the vehicle to the high risk of becoming immobilized in the obstacle because of mechanical failures. This method is also very costly in terms of the amount of time that is lost while conducting the self-breach. Only the road crater or the tank ditch may be self breached.

Now that the gap type obstacle representation has been briefly outlined, it is time to discuss the synergistic effects created by the emplacement of these obstacles on the battlefield. As in the previous discussion of the minefields, the actions will be discussed with respect to flowcharts included with the text. (See Figure 17)
1. **Gap Decision Ellipse Actions**

As the reader may recall, a decision ellipse can be entered in any of three ways, toward the attack on a route, coming back from an obstacle and from the side, when an entity is moving laterally in search of a bypass route.

An entity entering a gap decision ellipse from the flank has its attribute (FORMACODE), that initiated the lateral movement, reset to allow for future dynamic route selections. If this decision ellipse is on a route that circumvents the obstacle in question, then entities entering this particular decision ellipse are given the formation dictated by the bypass route and sent on this route toward their objective. Dismounted soldiers are not affected by gap obstacles and are allowed to proceed directly through them. If the entity enters the decision ellipse on the way back from an uneventful breaching attempt, then his rearward movement is altered so that pseudo dynamic route selection (bypass/lane) may occur. At this juncture, the knowledge link to the obstacle is performed for all entering entities regardless of the direction of travel upon entry.

Prior to continuing, a brief review of the order of preference for tactical responses will be listed. If a unit
Figure 17: Routine GAP-DECISION Flowchart
possesses the appropriate breaching equipment to counter an obstacle, then it will always attempt to breach the obstacle. If this is not the case and the "bull through" is not specified by the user on input, then a lane select or a bypass option will occur.

When an entity enters a decision ellipse and has no knowledge of the obstacle's existence, then the entity is sent directly toward the obstacle to continue the attack. If, however, the unit is aware of the obstacle's existence, (the knowledge level is two), then the unit is checked for possession of the appropriate breaching equipment to counter the impending gap type obstacle. The application of a particular piece of breaching equipment against a particular obstacle was discussed earlier in this section. If the platoon possesses breaching equipment, then it is allowed to continue in the attack toward the obstacle. The next check made when the knowledge level is two is to determine if the unit will bypass or "bull through" the obstacle. This determination is necessary only when the unit does not have applicable, organic breaching assets to neutralize the obstacle to its front. If the "bull through"/self breach is specified on input, then the entity continues to move toward
the obstacle. If the bypass option is chosen, then the unit will obtain the Bypass Route and Movement Control Point from the field's attributes and then move to the Bypass Route and MCP. If information about a breached lane through an obstacle exists, (knowledge level of three), then a platoon without proper breaching equipment will proceed to the lane that already exists through the gap. If a unit has the necessary equipment to perform a successful breach, it will always attempt to breach a lane on the present route of movement.

The Decision Ellipse is the main control mechanism for the selection of tactical alternatives available to the unit commander. The next section of the gap barrier discussion pertains to actions at and in the gap obstacle field ellipse. (See Figure 18)

2. **Gap Entry Actions**

The first check made when an entity enters a gap obstacle is to determine if the obstacle is emplaced/active. If the obstacle is not active, then the entering entity continues its unaffected movement in the current direction. At this point, a check to determine if the entity is mounted is made. If the entity is not in a vehicle, then it is
Figure 18: Routine GAP.ENTRY Flowchart
allowed to proceed unscathed through the barrier. Again, dismounted entities are assumed to be unaffected by gap type obstacles. Upon completion of these initial checks, the knowledge level of the obstacle is ascertained. If the knowledge of the obstacle is nonexistent, then the level is now switched to two, the entity and all affected units now know the location of this obstacle. The reader is reminded that the gap type obstacle field perimeter is defined by the visual recognition distance to the obstacle. If the route of present travel is a cleared/breached lane across the gap, then the entity will change formation to a column and proceed to the crossing site. If this route does not contain a breach, then a check of the status of the platoon’s organic, breaching equipment is made. If the platoon possesses appropriate breaching equipment to counteract the obstacle, then the equipment is sent forward to gain a breach across the gap. The ongoing breach flag, located in array TRAP.CONTROL is set to the number of the platoon presently attempting the breach on this route (see Appendix A for more details). The remainder of the platoon is halted, placed in half defilade positions, and provides overwatching fire for the breaching operation. An event is
scheduled every fifteen seconds to update the overwatching entities on the progress of their platoon's breaching operation. This event is called HEAVY.JUNK and is discussed in detail in Section five, below. When a platoon possesses no organic breaching equipment, a test is made to determine if a bypass route exists for the entities on this route. If the unit can bypass, then the entity's platoon is turned around and sent back to the decision ellipse. If the unit cannot bypass, then it must "bull through". When this is the situation, a check is made to determine if a breaching operation is already underway on this route. If a breaching attempt is already in progress, the entity's platoon will stop, move into half defilade positions, and cover the other platoon's breaching operation by direct fire. When this occurs an event called GAP.JOCK is scheduled. (See Figure 19) If, however, no breach is underway, then the entity moves to the gap and attempts a self-breach.

3. **Event GAP.JOCK**

This event simulates a breaching operation status report for the overwatching fire entities. If the breaching operation is still in progress and the breaching/bridging equipment is alive, then the supporting entities continue
Figure 19: Event GAP.JOCK Flowchart
their covering fire role. If, however, the operation has stopped, regardless of success or failure, the checking entity starts moving toward the gap to either cross over the successful breach or attempt a self-breach.

This completes the discussion of the gap obstacle field entry actions and the gap internal actions will now be explained. (See Figure 20)

4. **Gap Internal Actions**

At the gap, the knowledge level of the obstacle can only have two possible values, two or three. The first, representing an obstacle without a breach and the second, an obstacle with a breach/span across the gap. If the obstacle is spanned/breached (knowledge level of three), and the route being travelled is the cleared path, then the entity changes its formation to a column and reduces its speed in order to move through/over the crossing site. When no breach exists, then the breaching equipment, the Blade Tank and AVLB, must be programmed to perform different actions upon arrival at the obstacle than the self-breach type of tracked vehicles.

In order to make the distinction between the types of breaching equipment, two checks are made. These checks
Figure 20: Routine GAP-INTERNAL Flowchart
allow the appropriate actions to take place depending on the type of obstacle encountered and the type of equipment available in the platoon encountering the obstacle. In making the distinction between the two categories of potential breachers, when there is no breach through the obstacle on the route of travel, a check is made to determine if the obstacle is a destroyed bridge (short-wet gap). This check is made because the only entity allowed to reach a blown bridge location that has not been spanned, is an AVLB. The Blown Bridge obstacle always has a bypass option; the "bull through" is not permitted. If the obstacle is a road crater or tank ditch, then a check is made to determine if the entity causing this internal action is a piece of breaching equipment, a dozer blade equipped tank or an AVLB. When the activating entity is not a breaching vehicle, a check is made to determine if the obstacle is a tank ditch and the knowledge level is three. If this is the case, then the entity is simulated as moving to the adjacent breach by drastically reducing its speed and directing this platoon element into a column formation. This speed and column change feigns this lateral movement; the vehicle actually stays on its current route. If the
obstacle had no breach, (knowledge level of two), then the entity must self-breach. In this situation the vehicle is halted, put in gun tube defilade to simulate being in or at the bottom of the ditch, and is prevented from firing due to its position in the ditch. The probability of immobilization during a self breach is Monte Carlo'ed to determine if the entity will become a mobility casualty in the future. If the entity was fortunate enough to avoid becoming stuck in the ditch, then the vehicle is scheduled to start to move after sufficient time has elapsed to simulate a gap penetration by self breaching. If the obstacle is breached in this manner, then an event WALL.BREACH is scheduled to occur after the allotted time has passed and the vehicle is taken out of its defilade posture and allowed to fire again. The purpose of WALL.BREACH is to update the knowledge level of the obstacle to three in order to indicate this route as a breached lane through the obstacle to follow-on units. (see figure 21)

This completes the description of the internal action with a breach not in existence on the route of travel and the platoon void of any breaching equipment.
Figure 21: Event WALL.BREACH Flowchart
The discussion continues with the situation when the knowledge level is two and the platoon has breaching equipment on hand. For this discussion, the reader is again directed to figure 20. If the vehicle causing this internal action is a dozer blade equipped tank, then the tank is halted, put into half defilade while it fills in the excavation, and its direct fire capability is removed. When the breaching device is an AVLB, the bridge vehicle is stopped on the friendly side of the gap. In both cases an event GAP.BREACH is scheduled to occur after sufficient time has elapsed to simulate the appropriate breach/span. This event will be discussed in section 6 below. The possibility exists that an earlier breach was scheduled for this route. In order to simulate this situation, an event called HEAVY.JUNK is scheduled to update the breaching progress status of the obstacle.

The discussion has covered all the actions that can take place at the gap. At this point two events, HEAVY.JUNK and GAP.BREACH have been scheduled. These events and their ramifications with respect to the simulation will now be explained. (See Figure 22)
5. **Event HEAVY.JUNK**

   Event HEAVY.JUNK is scheduled by the platoon overwatching elements at field entry and again scheduled by the breaching equipment at the gap. This event serves as a situation report on the progress of the breaching effort. If the breaching effort is continuing, the platoon breaching equipment is checked. If the appropriate breaching equipment to counter the obstacle in question is still alive, then another status report, HEAVY.JUNK, is scheduled to take place in fifteen seconds. If, however, the obstacle is not breached and the platoon effort has been exhausted, then the array TRAP.CONTROL (see Appendix A) is checked to determine if the platoon can bypass this obstacle. If the platoon can bypass the gap, then the remaining unit members are turned around and sent back to the decision ellipse where additional platoon movement actions will take place. However, if the platoon cannot bypass, then the elements have to attempt a self-breach of the obstacle. The platoon elements covering the breach are taken out of their half defilade positions and moved toward the gap where the self breach will occur.
Figure 22: Event HEAVY.JUNK Flowchart
The remaining segment of Event HEAVY.JUNK pertains to the breaching equipment attempting to breach the gap. If the knowledge level is three, a breach is completed. If this route contains the breached lane, then the platoon is restarted. These entities are taken out of any defilade positions, assembled into platoon column and move in the direction of the attack. The breaching equipment departing the gap is slowed down just enough to let the remaining platoon elements catch up. During Event HEAVY.JUNK, anytime movement is initiated toward the gap, the TRAF.CONTROL flag signifying a breach in progress is turned off by setting the value to zero. (See Appendix A)

The last series of checks and actions for breaching equipment at the obstacle requires additional explanation. This is accomplished by an example. (See Figure 23)

At time 9:00, Platoon A arrived at the tank ditch in question and began a breaching operation. Two tanks are covering by fire and the platoon blade tank is starting to fill in the gap. The successful breach is scheduled to occur in six minutes, at 9:06. Platoon B is approaching the obstacle. (See Figure 24)
At 9:02 Platoon B arrives at the obstacle on the same route and schedules a breach to occur for its blade tank at 9:08.

<table>
<thead>
<tr>
<th>Platoon</th>
<th>Arrived</th>
<th>Breach Scheduled</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9:00</td>
<td>9:06</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Figure 23: Multiple Breach of a Tank Ditch, Time: 9:00

| X | is platoon A covering fire element. |
| (I) | is platoon A breaching equipment. |
| (S) | is platoon B covering fire element. |
| (#) | is platoon B breaching equipment. |

Platoon B's two other tanks remain at the field boundary and cover the breaching operation by direct fire.

At time 9:06, Platoon A completes a successful breach. Route 1 is now a cleared lane and the elements of Platoon A begin to move to cross the gap. As a result of the successful breaching operation by Platoon A, it is not
necessary for Platoon B to continue its breaching operation. Through the use of the event HEAVY.JUNK, Platoon B is also started up and allowed to cross the obstacle at approximately 9:06.

<table>
<thead>
<tr>
<th>Platoon</th>
<th>Arrived</th>
<th>Breach Scheduled</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9:00</td>
<td>9:06</td>
</tr>
<tr>
<td>B</td>
<td>9:02</td>
<td>9:08</td>
</tr>
</tbody>
</table>

![Diagram](image)

- X is Platoon A covering fire element.
- (X) is Platoon A breaching equipment.
- ¥ is Platoon B covering fire element.
- (¥) is Platoon B breaching equipment.

Figure 24: Multiple Breach of a Tank Ditch, Time: 9:02

This concludes the discussion of Event HEAVY.JUNK, the next explanation concerns the Event GAP.BREACH. (See Figure 25)

6. **Event GAP.BREACH**

Event GAP.BREACH simulates the actual break-through at the gap obstacle. At the beginning of this event, some
Figure 25: Event GAP.BREACH Flowchart
checks are made to ensure that the scheduling entity is still breaching and alive, and to determine the type of breaching equipment involved. If the entity is alive, breaching, and an AVLB, then it completes the bridge launching operation. At this time, the bridge launcher vehicle is put in full defilade and the AVLB as a participating entity is removed from the simulation. The assumption is made, that once a bridge is emplaced over a gap, the launcher remains at the bridge site and follow-on elements are allowed to cross over the gap on this bridge. Irregardless, of the type of equipment performing the breach, the remaining platoon elements are put into a column formation and movement is started to or from the gap depending on the element's present location. In addition, all vehicles are taken out of any defilade postures, the breaching equipment is allowed to shoot again, and is slowed to allow other (covering fire) platoon elements time to catch up. The knowledge level of the obstacle is changed to three to signify that this route has a breach. The breach in progress flag in TRAP.CONTROL is reset to zero (see Appendix A).
This concludes the discussion of the internal actions that occur at a gap obstacle. In the remaining section of this chapter, the actions taken upon exit from a gap obstacle field are discussed. (See Figure 26)

7. **Gap Exit Actions**

When an entity activates a gap obstacle field exit action, it is checked to determine if the entity is a vehicle. If the entity is a vehicle moving in the attack direction, it affects the knowledge level of the obstacle. Dismounted entities/soldiers are not allowed to affect the breach status of a gap obstacle. The status of the obstacle is again checked to determine if it is activated or just planned. If the entity is not a dismounted soldier and the obstacle is indeed a barrier to movement then the departing entity's speed is increased to the maximum allowed by the terrain and all vehicle attributes affected by the gap field are reinitialized. The last check determines if the entity is moving in the direction of the attack following a successful gap crossing or in the opposite direction, returning to the decision ellipse after an uneventful breaching attempt. If the latter is true, nothing further happens in the gap obstacle and the entity moves back to the
Figure 26: Routine GAP.LEAVE Flowchart
decision ellipse to select a lane or bypass the obstacle. If, however, the vehicle has passed through the crossing site and is continuing the attack, then the obstacle knowledge level is changed to three.

The synergistic effects of gap type obstacles have been described in detail. In the next and final chapter, the discussion will focus on some further enhancements to the Engineer Effects Module.
VII. FURTHER ENHANCEMENTS

The Engineer Effects Module for the STAR Combat Model has been discussed in detail. This chapter suggests some possibilities for future efforts to enhance the capabilities of this module. Many of these ideas are ones that time did not allow to be implemented in this first attempt at the module. Still, others were not thought of until the completion of the module and its associated documentation. In any event, there is still much to be done in the Engineer arena and these ideas may be helpful in future endeavors.

A. FURTHER CAPABILITIES OF THE OBSTACLES REPRESENTED

Scatterable minefields exist in the current module in both an activated or a deactivated state. This acts as an on and off switch for this obstacle. The Field Artillery could shoot scatterable mine rounds into an unactivated, preplanned, scatterable minefield ellipse and then activate this field by setting its attribute, P1.FLD, to the number of mines delivered in this field.

Most scatterable mines possess a self destruct mechanism. It is possible to simulate this characteristic
by scheduling an event to set the attribute PI.FLD of a scatterable minefield, to zero at some time in the future of the simulation. The time duration between the activation and deactivation time of the minefield should be the self destruct time of the mines in the field.

Further, the detonation of a scatterable mine could be affected by the probability of the mine being a dud, self destructing early, or not ejecting from the shell properly. For a more detailed discussion of these probabilities and effects, see Reference 28.

B. ENGINEER UNIT REPRESENTATION IN STAR

Engineer units are not currently represented in STAR. Now that the primary obstacles and their effects are modelled, a possible future iteration in the enhancement of the Engineer Module is the explicit modelling of Engineer units.

1. Engineers in the Offense

The discussion in Chapter two avoided the deliberate breach as an option for the tactical commander to utilize to overcome obstacles. If the Engineer units are specifically represented, then this tactical option becomes a reality on the simulated battlefield. Line charges and the Surface
Launched Unit Fuel Air Explosive system (SLUFAE), both organic to Engineer units could also be simulated. [Ref. 20]

2. **Engineers in the Defense**

Again, if the Engineer units are specifically represented, then their employment in conjunction with planned, inactivated obstacle ellipses to simulate obstacle construction, while the battle is ongoing could be a reality. This could be accomplished by planning to send Engineer units on routes through several unexecuted obstacle ellipses. Upon arrival at an inactivated obstacle field, an event to activate the obstacle could be scheduled to occur at some time in the future. The time until activation is the emplacement/excavation time required for the particular obstacle. This time parameter can be input as a field attribute. The Engineer units would be required to monitor the disposition of friendly forward elements. This is to determine if activation of the obstacle is still possible or if an early/uneventful field departure by the Engineers will occur. In any event, the Engineer unit would move to the next field obstacle ellipse planned on its route. Engineer units could either activate obstacles themselves or turn over prepared targets to tactical units for them to
detonate/activate. This cycle of events would continue until the Engineers reach their final defensive position.

C. EQUIPMENT ENHANCEMENTS/CAPABILITIES

The effectiveness of the mine roller or mine plow for clearing mines may be of some concern. If this is the case, the number of hits could be tallied in routine POP.A.MINE using the existing framework for determining the hit location of a mine in relation to the activating vehicle. The only dimensions required are the skid width of the mine plow or the roller width of the mine roller. In addition, a roller or mine plow could be eliminated after reaching some threshold value of mine hits or each hit could be Monte Carlo'ed against a probability of kill of a roller or a plow.

The Armored Vehicle Launched Bridges (AVLB) that span a gap are currently removed from the simulation as soon as the bridge is launched. If the user wishes to reactivate this employed bridge, then an event to change the launched status of the bridge vehicle would have to be scheduled to occur at some future time. This event would alter the value of a bridge vehicle's HV.STATE attribute, change its full defilade posture, and file it in new PLATOON, COMPANY, and
BATTALION sets. In addition, the breach status on a route through an obstacle would have to be changed to reflect the removal of the bridge from the gap.

D. ADDITIONAL OBSTACLE REPRESENTATION

A built up area could be represented by an ellipse that would slow and channelize movement. Incoming and outgoing direct fires from this ellipse could be modified or halted in order to simulate vehicles being masked by buildings in the built up area. This could be accomplished by changing the entity's defilade posture to halt incoming direct fire and either its plow or blade conditions to halt outgoing direct fire. Values of four or two, for PLOW.COND or BLADE.COND, respectively, would accomplish this effect.

A river crossing site ellipse could be used to simulate vehicles crossing a river by swimming or snorkeling. This could be the "bull through" option, near a blown bridge obstacle. Time could be assessed against tanks that had to affix snorkeling fixtures, their speed could be degraded, direct fire stopped, and they could be put in a full defilade posture (underwater). The amphibious vehicles would have similar actions taken to degrade their performance.
E. OTHER TACTICAL OPTIONS

The option to cover a deployed mine breaching vehicle by platoon or company direct fire should be added to this Engineer Module. The basic logic and structure for its implementation already exists in the gap obstacle portion of the module.

Imperfect or nonexistent intelligence between attacking echelons could be simulated by zeroing out the TRAP.CONTROL matrix and re-initializing the knowledge level of the obstacles, field attribute P6.FLD, between passage of the echelons.

The Engineer Effects Module currently simulates the following tactics.

1. Breach the obstacle if possible, otherwise "bull through".
2. Breach the obstacle if possible, otherwise select a clear lane if one exists, or bypass the obstacle.

Additional tactical options could be modeled with minor modifications to existing code. This may require the addition of another attribute to the obstacle ellipse to indicate the appropriate tactical option. Some additional tactical options include the following:

1. Select a clear lane if one exists, otherwise bypass, breach or "bull through" as a last resort.
2. Breach the obstacle if possible, otherwise select a clear lane if one exists, or "bull through" as a last resort.
7. MISCELLANEOUS

In the current Engineer Module, AVLB vehicles are prevented from firing by giving them no ammunition, however, they still search for targets. In the new Night Vision Lab Search Model, target selection by AVLB's can be turned off by defining a search type which calls for a search tactic of zero. A search tactic of zero does not exist and the simulation will only call one search for the bridges. However, the flash detection may be an exception to this rule. Thus, a search or detect will never occur for a bridge entity again. This would yield a small savings in computer time, as a result of fewer searches during the model run.

In this chapter, the discussion has centered around ideas to further enhance the capabilities of the Engineer Effects Module. In a modelling project of this magnitude, there is no fitting conclusion. As each piece of the Combat Modelling process is completed, more questions pertaining to the whole process are generated. Perhaps this is what makes Combat Modelling the iterative process it is. One statement seems to suffice...The effort goes on.
In this appendix, the dimensioning and function of the array TRAF.CONTROL is discussed. This array is the key to the implementation of pseudo-dynamic movement in the Engineer Effects Module for STAR. The array TRAF.CONTROL is accessed many times throughout the simulation to obtain information necessary for pseudo-dynamic movement. In order to provide the reader with a more complete picture of TRAF.CONTROL, it is described in detail in this appendix.

A. THE DIMENSIONING OF TRAF.CONTROL

TRAF.CONTROL is a three dimensional real array with the following subscripts:

OB: obstacle number (1,2,...OB.NUM)
RT: route number (1,2,...RT.OB.NUM)
K: integer index (1,2,3,4,5)

The number of obstacles (input parameter - OB.NUM) and the number of affected routes (input parameter - RT.OB.NUM) each play a critical role in the dimensioning of this array. There are several ways to obtain these input parameters,
some save computer time by demanding some input manipulation, while others use more computer time but are more user friendly. The more efficient example is described first.

1. If the user puts the obstacle ellipses first on the input list, then OB.NUM will be the smallest possible value. Since the first dimension of TRAF.CONTROL is based on OB.NUM this action makes the first dimension of TRAF.CONTROL the smallest it can be.

2. For the second dimension of TRAF.CONTROL, if the user takes the routes that are affected by the obstacles and puts them first on the input list, the RT.OB.NUM will be the smallest possible value. Since the second dimension of TRAF.CONTROL is based on RT.OB.NUM, this step makes the second dimension of TRAF.CONTROL also the smallest possible value.

3. The third dimension of TRAF.CONTROL is indexed from 1 to 5.

Since the above procedure provides TRAF.CONTROL with the smallest possible dimensions, it is the most efficient method of dimensioning TRAF.CONTROL. It is true that the SINSSCRIPT language possesses the capability to utilize
ragged arrays. The authors pondered over this very issue and made a conscious decision not to use this capability. Since TRAF.CONTROL is accessed so often in the Engineer Effects Module, the authors determined that direct access to its contents was more desirable than the numerous searches of array data required by the ragged array technique. Other alternatives to the manipulated route and obstacle approach listed above range from using just step number one above and not ordering the routes to using step number two and putting the fields in any order or finally, not ordering anything, thus creating a TRAF.CONTROL array that is very large and inefficient in terms of computer storage. In this last case, OB.NUM is the total number of fields used in the simulation and RT.OB.NUM is the highest route number used.

With the dimensioning of TRAF.CONTROL explained, the remaining discussion will focus on the contents of TRAF.CONTROL and its function in the Engineer Effects Module.

B. THE CONTENTS OF TRAF.CONTROL

TRAF.CONTROL contains information that is slightly different for the two major types of obstacles modeled. This section discusses the values with the obstacle
differences taken into consideration. The array TRAF.CONTROL is accessed by obstacle number, route number, and the third dimension value. The third dimension values in each of the five positions are the values that are required by the simulation. It is for this reason that the reader is reminded that the values being described in the five positions exist for each obstacle and associated route through the obstacle. The values that are stored in these five places are listed by major obstacle type below:

1. **Minefield TRAF.CONTROL Contents**

These are the items contained in each of the five positions of the array TRAF.CONTROL with regard to minefields.

1. **The route number, the entering entity is travelling on in the minefield.** This number is accessed by follow-on units when this is a cleared lane through the minefield.

2. **The Movement Control Point in the decision ellipse or a value of zero, if the entering entity is going to "bull through" the minefield.**

3. **The cumulative distance clear in the minefield on this route.** If this value is plus infinity, this route is a lane through the minefield.
4 The most recent distance to mine encounter for the lead vehicle in the lead platoon trying to traverse the minefield.

5 The total number of mine encounters scheduled thus far for the elements travelling on this route through the minefield. (bump number)

2. **Gap Obstacle TRAP.CONTROL Contents**

These are the items contained in each of the five positions of the array TRAP.CONTROL with regard to gap obstacles.

1. The route number that the activating entity is travelling upon entry of the gap obstacle ellipse. This number is accessed by follow on units when this route has a breach/span through the obstacle.

2. The Movement Control Point in the decision ellipse or a value of zero, if the entering entity is going to "bull through".

3. The lane status of this obstacle for a particular route.
   - 0 - a lane does not exist on this route.
   - + infinity - a lane exists on this route through the obstacle.
The breaching status for this route. If the value is 0, then there is no breach in progress on the route. Any other number indicates the platoon number of the platoon currently attempting a breach on the route.

C. THE FUNCTION OF TRAF.CONTROL

The values stored in this array differ for the major obstacle types, thus, the discussion is categorized in this manner.

1. TRAF.CONTROL Function in Minefields

The first position in TRAF.CONTROL contains the route number of the entity travelling in the minefield. This route number is entered in TRAF.CONTROL upon entry into a minefield by an entity. The second position pertains to the tactical option to "bull through" or to bypass the minefield. The options to "bull through" or to be able to bypass, are values specified as attributes by the user. These values determine what value will be placed into TRAF.CONTROL position number two. If the user specifies that the "bull through" option is to be used, i.e. there is no bypass route, then the entity conveys a value of zero to
this position upon entry into a decision ellipse. If the entity has the option to bypass the minefield, then the entity conveys the number of the Movement Control Point in the decision ellipse to position number two. At this point the simulation has all the necessary information to perform pseudo-dynamic route selection of cleared lanes in the minefield. If a clear lane through the minefield exists, and entities have decision ellipses and the bypass option, then the TRAF.CONTROL array is searched over position three to locate a lane. The affected units are then redirected to the clear route and MCP, stored in array TRAF.CONTROL for the obstacle.

The remainder of the discussion of TRAF.CONTROL pertaining to minefields concerns positions three, four, and five. Each vehicle (entity) has an attribute called the tank bump number (TBUMP.NUM). This attribute allows the entity to move as a member of a platoon when in a minefield. Upon entry into a minefield the value of TBUMP.NUM is always zero. If the bump number (position number five) in TRAF.CONTROL is greater than the TBUMP.NUM of the vehicle entering the field, then another vehicle has preceded it into the field. Since another vehicle is already some
cleared distance in the minefield, the entering vehicle is allowed to travel that distance with no mine effects. In order to implement this effect, TRAP.CONTROL is utilized further.

The difference between the TBUMP.NUM and the bump number stored in TRAP.CONTROL is computed. If the difference is one, the entering vehicle receives the most recent distance to a mine encounter for the platoon lead vehicle. This value is stored in position four of TRAP.CONTROL. If the difference is more than one, the vehicle's next mine encounter distance is set to the cumulative distance clear for this route and stored in the TRAP.CONTROL matrix, position three. The vehicle's TBUMP.NUM is updated to the value of the TRAP.CONTROL bump number. If, however, the difference between TBUMP.NUM and the bump number in position five is zero, then the vehicle is the first from the platoon to enter the minefield. Routine MINE.SCHEDULE is called and a distance to a mine encounter is computed. This distance value is placed in TRAP.CONTROL positions three and four and both the array bump number (position five) and the vehicle's TBUMP.NUM are set to one.
The use of the bump numbers at the minefield internal actions is nearly the same as in the discussion above. The difference between the vehicles TBUMP.NUM and the value of the TRAF.CONTROL array bump number is calculated. If the difference is zero, then this entity is the lead element in the platoon. The next mine encounter distance is obtained from routine NINE.SCHEDULE and this distance is placed in position four of TRAF.CONTROL. This distance is then added to the value (cumulative distance clear on the route) already stored in TRAF.CONTROL position three. The entity's TBUMP.NUM and the array bump number are both incremented by one. If on the other hand, the TBUMP.NUM of the activating entity is less than the bump number (position five) of the TRAF.CONTROL matrix, then other actions take place. If the difference is only one, then the distance stored in position four of TRAF.CONTROL will be transmitted to the activating entity as his next mine encounter distance and the entity's TBUMP.NUM is increased by one. If the difference is more than one mine encounter distance, then the distance from the activating entity to the lead platoon vehicle's projected cleared location is calculated and this becomes the distance to the
next mine encounter for the activating entity. This entity’s TBUMP.NUM is given the value of the array’s bump number (position five). This process continues until the platoon traverses this minefield or is depleted. When an entity exits a minefield after a successful breach, then the cumulative distance clear value for the route, TRAP.CONTROL storage location three, is set to plus infinity. This signifies that a breach exists through the minefield on this route and allows follow-on units to search TRAP.CONTROL to find the cleared lane for this obstacle.

2. TRAP.CONTROL Functions In Gap Obstacles

The same actions for the TRAP.CONTROL array for positions number one and two occur for the gap type obstacles. The storage for positions three, four and five differ from the previous discussion concerning minefield interactions. Position number five is not used by entities involved in gap obstacles. The third storage location is used as a flag to trigger dynamic lane selection. If it contains a value of plus infinity, then a successful breach has been performed through this tank ditch and follow-on unit movement is redirected toward this lane. If the value contained in position three is zero, then the gap has not
been penetrated and units must either attempt a self breach, breach with equipment, or bypass this excavated obstacle. TRAF.CONTROL position number four contains the number of the platoon that is already performing a breaching operation on this route at the gap. This information gives follow-on platoons, that would otherwise have to attempt a self breach, the opportunity to cover the breaching platoon's effort by fire. In addition, if the breach attempt is successful, the follow-on platoon can utilize the breach gained for their crossing of the gap.

The discussion of the TRAF.CONTROL array is, indeed an important one because this array organizes the movement of platoons and enables units to dynamically select cleared lanes. In addition, TRAF.CONTROL coordinates fire support of current breaching efforts from follow-on units on the same route in the instance of a gap obstacle. This concludes the explanation of the array TRAF.CONTROL.
APPENDIX B

MODIFIED ROUTINES AND EVENTS FROM BASIC STAR

In the construction of this model, many changes were made to the existing STAR code. In order to facilitate the location of these changes the authors have supplied several lines of unaffected code occurring before and after the changed lines. In addition, the lines with changes are further highlighted through the use of a plus sign (+) immediately following the line number.

1. PREAMBLE

Purpose

The PREAMBLE provides the compiler with definitions regarding entities, attributes, and sets; events and routines; background mode, type and dimensionality and global variables and arrays.

Modifications

The following additions have been made to the PREAMBLE. For additional information refer to the thesis section listed below.
Events (Appendix C)

- DITCH
- KILL
- DIVERT
- GAP
- BREACH
- GAP
- JOCK
- HALT
- HEAVY
- JUNK
- QUIK
- MOVE
- STAND
- TO
- TURN
- AROUND
- WALL
- BREACH

Global Variable

FLD.POINTER (1-D) Integer

This 1-dimensional array contains the temporary field pointers. This enables the user to access the obstacles by using the sequential order numbers given the fields on input.

TRAF.CONTROL (3-D) Real (Appendix A)

Permanent Attributes Integer

GRUNTLETH. (3-D)

This 3-Dimensional array contains the BRL casualty data for anti-personnel mines.

MINELETH (4-D)

This 4-Dimensional array contains the BRL casualty data for anti-tank mines.

PLT.COND (1-D)

This attribute of the platoon leader keeps track of the present action status of the platoon in a minefield (i.e. plowing, offsetting, pushing, etc.)

160
NXT.BELT  (1-D)

This platoon leader attribute stores the number of the next mine belt to be encountered in the minefield. It allows the platoon to move through a minefield as an organized unit.

Temporary Attribute  Integer

BLADE.COND

This unit attribute gives the tank the ability to have a dozer blade characteristic.

FLD.FORM

This unit attribute gives the entity the ability to change formations when encountering an obstacle.

HOLD.FORM

This unit attribute is a storage place, where the units formation number for the platoon (FORMACODE) can be placed when lateral route movement is appropriate.

PLOW.COND

This unit attribute gives the tank the ability to have a mine plow characteristic.

TBUMP.NUM

This unit attribute allows the entity to move in a very detailed manner as a member of his platoon when in an obstacle field.
TPNAM.FLD

This field attribute keeps the temporary pointer number of the field.

WHAT.BELT

This unit attribute works in conjunction with the platoon leader attribute (NXT.BELT).

In order for an event to have arguments passed from event to event or routine to event, etc., the arguments must to be given in the PREAMBLE.

The following arguments will be described in Appendix C.

BAGGED.BOY
BREACHER
BYPASS
GULLY.CAT
HOLE
HOME
INT.TIME
NUM.PONTER
RAMMER
REP.NUMBER
SABOT.SHOOTER
SIDE.STEPPE
SLOW.DOWN
STATUS
TEAR.DOWN
TWIST

Temporary Attribute Real

ANGLE.FLD

This field attribute carries the orientation angle in radians measured counterclockwise from east to the major axis.
AREA.FLD

This field attribute carries the area of the field in meters squared.

FSPEED.FAC

This unit attribute gives the entity the ability to slow down because of obstacle encounter.

P5.FLD (Appendix D)
P6.FLD (Appendix D)
P7.FLD (Appendix D)
P8.FLD (Appendix D)
P9.FLD (Appendix D)
P10.FLD (Appendix D)
P11.FLD (Appendix D)
P12.FLD (Appendix D)

SAMAJ.FLD

This field attribute is the semi-major axis length of the elliptical field in meters.

SAMIN.FLD

This field attribute is the semi-minor axis length of the elliptical field in meters.
Brief Explanation

The additions are shown as they appear in the code. Enough of the basic STAR code is given so that the additions can be quickly added. As mentioned previously, a + sign immediately after the line number highlights the lines that have been changed. This procedure will be used throughout the remainder of this appendix to annotate changed basic STAR code.

CODE

```
1 PREAMBLE
12 THE SYSTEM HAS A LE6D(*/4), A LE5D(*/4), A DLA(*/4),
13 A MVARY(*/4), A ARRAY(*/4), A PHSCORD(*/4),
14+ A BN.AIR.PRI(*/4),
15+ A BNCORD(*/4), A COCORD(*/4), A RTAB(*/4),
16+ A GRUNTLETH(*/4), A LE12(*/4), A LE72(*/4),
17+ A LE31(*/4),
18+ A LE81(*/4), A LE83(*/4), A LE11(*/4),
19+ A LE61(*/4), A LE81(*/4), A LE83(*/4), A LE11(*/4),
20+ A GRUNTLETH(*/4), LE12(*/4), LE72(*/4),
21+ A LE31(*/4), LE81(*/4), LE83(*/4), LE11(*/4),
22+ A BN.AIR.PRI(*/4),
23+ A BNCORD(*/4), A COCORD(*/4), A RTAB(*/4),
24+ A LE81(*/4), A LE83(*/4), A LE11(*/4),
26+ A GRUNTLETH(*/4), LE12(*/4), LE72(*/4),
27+ A LE31(*/4), LE81(*/4), LE83(*/4), LE11(*/4),
28+ A BN.AIR.PRI(*/4),
29+ A BNCORD(*/4), A COCORD(*/4), A RTAB(*/4),
30+ A LE81(*/4), A LE83(*/4), A LE11(*/4),
31+ A LE61(*/4), A LE81(*/4), A LE83(*/4), A LE11(*/4),
32+ DEFINE BNCORD, COCORD, TABLE, RTAB, USMG, GRUNTLETH,
33+ DATA, SP, POINT, HOLD
34+ AS INTEGER 3-DIMENSIONAL ARRAYS
35+ DEFINE SOVKG, HINLETH AS INTEGER 4-DIMENSIONAL ARRAYS
36+ Defines BNCORD, COCORD, TABLE, RTAB, USMG, GRUNTLETH,
37+ DATA, SP, POINT, HOLD
38+ as integer 3-dimensional arrays
39+ defines SOVKG, HINLETH as integer 4-dimensional arrays
40+ SEVERAL OWNS FOLLOW
41+ SEVERAL OWNS FOLLOW
42+ EVERY SQUAD.LEADER OWNS A SQD.UNIT
43+ EVERY PLATOON.LEADER OWNS A (R4HAT(1-15),R3HAT(16-32)),
44+ A (R2HAT(1-15),R1HAT(16-32)), A (R3HAT(1-30),GONE(31-32)),
45+ A NIT.BELT, A PLT.COMD
46+ AND OWNS A PLT.UNIT
47+ EVERY COMPANY.COMMANDER OWNS A P.TO.ARRAS,
48+ A (CORT(1-5),CROST(6-7)),
49+ A CONNECTS AND MAY BELONG
50+ TO A BATTALION
51+ EVERY BN.COMMANDER OWNS A (BNWRT(1-5),BNCUR(6-11),
52+ BNSLO(12-17),BNSGO(18-23)),
53+ BATT(24-30),BROST(31-32)),
54+ OWNS A BATTALION AND
55+ MAY BELONG TO A BRIGADE
56+ EVERY BDE.COMMANDER OWNS A (BDECUR(1-10),BDEL0(11-20),
57+ BDEBGO(21-32)),
58+ MAY BELONG TO A DIVISION
59+ OWNS A BRIGADE, AND MAY BELONG TO A DIVISION
60+ Defines S1DED,S2DED,S3DED,S4DED,S5DED,S6DED,HV.UNIT,
61+ Defines S1DED, S2DED, S3DED, S4DED, S5DED, S6DED, HV.UNIT,
62+ Defines S1DED, S2DED, S3DED, S4DED, S5DED, S6DED, HV.UNIT,
63+ Defines S1DED, S2DED, S3DED, S4DED, S5DED, S6DED, HV.UNIT,
64+ Defines S1DED, S2DED, S3DED, S4DED, S5DED, S6DED, HV.UNIT,
65+ Defines S1DED, S2DED, S3DED, S4DED, S5DED, S6DED, HV.UNIT,
66+ Defines S1DED, S2DED, S3DED, S4DED, S5DED, S6DED, HV.UNIT,
67+ Defines S1DED, S2DED, S3DED, S4DED, S5DED, S6DED, HV.UNIT,
CUDEDS1SRT, S2SRT;
S3SRT, S4SRT, S5SRT, S6SRT, CSURRT, PRTY, REDSIN, REDDRG,
BLUESIN, PRTY, CNDT;
R1DED, R2DED, R3DED, R4DED, R5DED, R6DED, RV.UNIT, R6DED,
R1DED, R2DED, R3SRT;
R2HAT, R3HAT, R4SRT, R5SRT, R6SRT, RUSRT, XCENT, YCENT,
R1HAT, R2HAT;
BNCUR, BNLO, BNGO, BAT1, BREQST, BDECUR, BDELO, BDEGO
AS INTEGER VARIABLES
EVERY DIV. COMMANDER HAS A DIVWT, A DIVCUR, A DIVLO,
A DIVGO, A DIVT, A DIVF, A DIVREQST, A NO.DIV.Unit,
AND OWNS A DIVISION

";

TEMPORARY ENTITIES

GENERATE LIST ROUTINES

EVERY FIELD HAS A NAME, FLD, A XC.FLD, A YC.FLD,
A PX.FLD, A TP.FLD, A ANGLE.FLD, A SAMAJ.FLD,
A SAMIN.FLD, A NAM.FLD, A XTPAM.FLD, A TP1.FLD, A TP2.FLD,
A P9.FLD, A P10.FLD,
A P11.FLD, A P12.FLD AND MAY BELONG TO A FLD.SET

DEFINE XC.FLD, YC.FLD, PX.FLD, TP.FLD, ANGLE.FLD, SAMAJ.FLD,
AREA.FLD, P1.FLD, P2.FLD, P3.FLD, P4.FLD, P5.FLD,

DEFINE XTPAM.FLD, NAM.FLD, TPAM.FLD AS INTEGER VARIABLES

EVERY UNIT HAS A NAME(1-11), COLOR(12-12),
SYS.TYPE(13-16), WPN.TYPE(17-23),

VERTYPE(24-27), ALIVE.DEAD(28-29), DEPNUM(30-32),
A (KHZ.LAST.TBTG(1-24), C2(25-30)), A ROUND,
A (ND.HIT(1-6), M.HIT(5-7), F.HIT(8-10), MP.HIT(11-14), K.HIT(15-17),
NUM.HIT(18-22),
FIRED.AT(23-27), KILL(28-28), FILL(29-29),
KILL(30-30), KILL(31-31),
A (SCLEAD(1-24), SJOVEN(1-24), CAR(25-32),
A (1-24), A MICRO.A HIT, STATE.
A (SBOC(1-10), PLT(11-18), CO(19-23), BM(24-28),
FID(29-30), SCHED(31-32),
A (RG(1-5), L.HAT(6-9), SMX.DET(9-11),
SMX.UNIT(12-14), CMX(15-16),
A (H.BLUE.LIVE(16-16), H.BED.LIVE(17-17),
B.UNIT(18-18),
H.UNIT(19-19), H.TANKS(20-20), HITSHOT(21-23),
H.UNIT(24-24),
PROJO(28-32), A (COCDE(1-24), PH(25-27),
CFP(28-32),
A (PTLDER(1-24), TRF(25-32), A (CARR(1-4),
H.COM(5-7), BNXK, NPC(6-14),
CP.WATT(1-18), POS.POINT(15-19), ROUTE(20-20),
I.ROUTE(21-28),
FORHACODE(29-32), A SRD(1-16),
AP.TOT(26-32), A (HE.DRAG(1-8),
AHHO2(1-8),
AW1.OR.BEL3(9-20), AHHO3(9-20), ANH3(21-23),
AW1.OR.BEL2(21-32), A
D.O.F.UNIT, A PRI.DIR, A LST.DIR, A X.CURRENT,
A Y.CURBENT
A Z.CURRENT, A M.D, A P.D, A MP.D, A TI.M.FIRE,
A (POE(1-24)
C.1(25-30), TASKED(31-31)), A (POS.IN.PLT.AREA(1-6),
FORMPOS(7-22), END.AREA(23-32)), A (FON(1-1),
DISC.ON.BT(2-2),
R.D.STATUS(3-4), SCH.TO.MOVE(5-6), GRNDATK(7-8),
FIL.MODE(9-9),
N.HD(15-19), N.AB(20-25), N.TH(26-32)), A (FIL.D.NO,
A (WHAT.BELT(1-10), FLD.AKT(11-16), N.PLE(17-23),
N.ARM(24-32)),
A (PA(1-8), ABT.HITS(9-18), FLD.FORMA(19-25),
TBMP.NUM(26-32)),
A (SLOW.COND(1-8), HLD.FORMA(9-19),
BLADE.COND(20-32),
A (SPEED.FAC, A P.BODY.INCAP, A P.HD.INCAP,
A P. TH.INCAP, A P.RE.INCAP, A P. AB.INCAP, A P.PLE.INCAP, A FIL.D.INT.DIST,
A FIL.BDY.DIST,
A POP.UP.TIME, ' ' USED FOR COMING OUT OF
FOXHOLE AFTER FA IMPACT
AN PUTANK.LIST, ' ' FOR COMMO
A (PROE(1-10), N.OBS(6-9), N.RDR(9-9),
THERMAL(10-10), ' ' NW(11-11),
RADAR(12-12), POP.UP(13-13), BASE(14-20), LSS(21-21),
LD(22-22), LDH(23-23), LSHO(24-24), A (CURR.SORTIE(1-24),
CP.FINAL(25-25),
AN (ORB. BT(1-24), FHR(25-30), EDR.PAINT(31-32),
A (BURS. STRIP(1-4), N.BURSTSA(5-8), REM. LO.CT(9-19),
DET. SEC(20-20), DET.PLT(21-21),
SEL. ARC. SEL. PLT(22-23), SEL. PLT(23-23), SN. ABO.C(24-24),
LWD(25-25), A PBB,
A GUIDE. OFF. AN EX. TIME, A (PASSOFF(1-24),
NO.FIRE(25-25)),
AND NO. BELONG TO A TANKS, A BLUE.ALIVE,
A RED.ALIVE, A COMP. UNIT,
A P.LT.UNT AND A SQD.UNIT
A INHIBIT LIST ROUTINES
DEFINIE BASE.COLOR, SYS.TYPE, PN.T. TYPE, VEH.T. TYPE,
ALIVE.DEAD, DEBD. THUS,
HIT LAST.TG.T, C.2, ROUND, MD. HIT, M. HIT, P. HIT,
MP. HIT, K. HIT, NUM. HIT,
FIRED. AT, KILL, KILL, KILL, SECLDR, CBAR,
SEC. PLT, CP.BW, IP.
Scheid. HIT, HIT.BAT, HIT.BATX, DET, MD. V. STATE, CSS,
R. BLUE, ALIVE, R. RED, ALIVE,
H. COMP.UNIT, H. PLT.UNIT, H. TANKS, HITSHOT, MISSSHOT,
PROJ. BLOW, COND,
COC.B, CH. CRP, PLT.LDR, TF. CARR, R. CON., NEXT. MSCP,
CP. NEXT, WHAT. BELT,
POS. PRNT, ROUTE, J. ROUTE, FORMACODE, RANGE, AREA,
VAR. AP. TOW. AP. NO.1,
HE.DRAG.AMEO2, AS.I. OR. ASL3, AMM93, AMO2. OR. AMO2, AMMO4,
POE, C.1. TKED,
POS.IN.PLT.AREA, FORMPOS, START. AREA, END. AREA, FON,
DISC.ON.BT,
FIL.FORMA.TBMP.NUM, BLADE.COND, HLD.FORM. AS INTEGER
VARIABLES
' ' INFRANTRY INTEGER VARIABLES FOR UNITS
DEFINIE P. NO, P.LT.AKT, P.ABB, N.TH, N.HD, N.AB, N.PLE,
P.TE. MODE, GRNDATK,
SCH.TO.MOVE, SQD.VEH AND R. D. STATUS AS INTEGER
VARIABLES
2. **Routine BL.CRE ATE**

**Purpose**

Routine BL.CRE ATE is used to create temporary tank entities at designated times throughout the execution of the program. The routine is called during the execution of Event BLU.FORCES.
Modifications

The following four attributes must be assigned values from the input data or as a result of the routine.

Temporary Attributes Integer

BLADE.COND
HOLD.FORM
PLOW.COND

Temporary Attribute Real

FSPEED.FAC

Brief Explanation

Line 15  PLOW.COND and BLADE.COND are assigned the values of 0-no or 1-attached.

Line 18  FSPEED.FAC is initialized to 1.0. No effect unless in field.

Line 19  HOLD.FORM is initialized to 999. This value changes only when the entity is using lateral movement. When movement is terminated, this 999 value is set back on HOLD.FORM for the entity. It serves as a flag to determine appropriate actions. To change this initial value has unpredictable consequences.
CODE

1 ROUTINE BL.CREAE(A,B,N)
11 CREATE A TANK
12 LET PL.HAT(TANK)=1
13 READ NAME(TANK),SYS.TYPE(TANK),WPN.TYPE(TANK),
14 SEC(TANK),PLT(TANK),CO(TANK),
15 BN(TANK),SODVER(TANK),PL.TLDR(TANK),COCOR(TANK),
16 START.AREA(TANK),
17+ ALIVE.DEAD(TANK),FIR.MODE(TANK),PLOW.COND(TANK),
18 BLADE.COND(TANK)
19 CALL FLD.DIST(TANK)
20 LET FLD.INT.DIST(TANK)=RINF.C
21 LET PSPEED.PAC(TANK)=1.0
22 LET HOLD.FORM(TANK)=999
23 IF CO(TANK) GT COS LET COS=CO(TANK) ALWAYS
25 RETURN END

3. Routine MOVE

Purpose

Routine MOVE is called from routine LOC to update the location of any entity to the current time of the simulation.

Modifications

The following entity attribute has been added to the MOVE routine logic. This addition now gives the user the ability to change formations as a result of encountering an obstacle field. The capability to offset around a dead vehicle is now a reality.

Temporary Attribute Integer

FLD.FORM
Brief Explanation

Lines 118, 160-165, and 195-200

Allow the FLD.FORM attribute the ability to override the normal formation procedure.

CODE

1 ROUTINE TO MOVE GIVEN VEH
78 'NEW.MCP'
82 LET MCP = NEXT.MCP(VEH) LET MM = MCP * 3
83 IF MCP EQUALS 0 "MOVE TO POSITION IN END.AREA
84 IF POS.IN.PLT.AREA(VEH) EQUALS 0, CALL
BEST.POS(VEH)
85 ALWAYS
86 LET I = PLT(VEH) LET K = POS.IN.PLT.AREA(VEH) * 3
87 FOR J = 1 TO DIR.F(POSITION(I,K)) WITH
POSITION(I,J) EQUALS
END.AREA(VEH)
88 DO
89 LET X.DEST = POSITION(I,J,K-1)
90 LET Y.DEST = POSITION(I,J,K)
91 LET DIR = POSITION(I,J,K+1)
92 LOOP
93 LET D.TO.MCP = SQRT.(X.DEST - X.CURRENT(VEH))**2 +
(Y.DEST - Y.CURRENT(VEH))**2)
94 IF D.TO.MCP LESS THAN ZERO.LEVEL,
95 LET NV.STATE(VEH) = 4
96 LET DIR.OF.NVRT(VEH) = DIR
97 LET SPD(VEH) = 0.
98 LET FINAL = 1.
99 GO TO NEW.INCR
100 ELSE
101 GO TO DRTN.COMP
102 ELSE IF FORMACODE(VEH) = 0 AND FLD.FORM(VEH) = 0
103 "GO DIRECTLY TO NEXT MCP
104 LET I.DEST = RTE.DATA(RT.MK-1)
105 LET D.TO.MCP = SQRT.(X.DEST - X.CURRENT(VEH))**2 +
(Y.DEST - Y.CURRENT(VEH))**2)
106 GO TO DRTN.COMP
107 ELSE "MOVE ALONG ROUTE OFFSET BY FORMATION
108 IF MCP EQUALS K/3 AND D.ON.VEH EQUALS 1
109 LET MK = RTE.DATA(RT.K-5)
110 LET NY = RTE.DATA(RT.K-4)
111 LET LY = RTE.DATA(RT.K-3)
112 LET I = RTE.DATA(RT.K-3)
113 ELSE
114 GO TO INTERMID
115 ALWAYS
116 LET NLX = NL.LK
117 LET NLY = NL.LY
118 IF FLD.FORM(VEH) NE 0
119 LET I = FLD.FORM(VEH)
Routine MOVE.LIMITS

4. Routine MOVE.LIMITS

Purpose

Routine MOVE.LIMITS is called from routine MOVE to determine a vehicle's maximum speed, acceleration rate, and deceleration rate. These parameters are a function of the weapon type, the system type, the slope of the terrain and the dismounted tactical situation.

Modifications

The routine now returns speed limits that are also a function of an obstacle encounter speed degradation.
Temporary Attribute Real

FSPEED.FAC

Brief Explanation

Line 18 The effect (degraded speed) of FSPEED.FAC has been introduced here. FSPEED.FAC may have any value between 0 and 1.

CODE

1 ROUTINE MOVE.LIMITS GIVEN VEH,SLOPE YIELDING SPEED,
2 ACCEL, DECEL
3 "" MOVE.LIMITS LOOKS UP LIMITING SPEED AND
4 ACCELERATION VALUES BASED ON
5 "" SYSTEM TYPE, WEAPON TYPE, AND TERRAIN SLOPE
6 IF SLOPE LE Lim.SPDS(SYS,WPN,2)
7 LET OFFSET = 2 "" DOWN SLOPE
8 ELSE
9 LET OFFSET = 1 "" LEVEL
10 ALWAYS
11 ALWAYS
12 LET SPEED = Lim.SPDS(SYS,WPN,3+OFFSET)*FSPEED.FAC(VEH)
13 LET ACCEL = Lim.SPDS(SYS,WPN,6+OFFSET)
14 LET DECEL = Lim.SPDS(SYS,WPN,9+OFFSET)
15 END

5. Routine RED.CREATE

Purpose

This routine serves the same purpose as does BL.CREATE except it acts for the Red (opposing) force.

Modifications

The following four attributes must be initialized:

(172
Temporary Attributes  Integer

BLADE.COND
HOLD.FORM
PLOW.COND

Temporary Attribute  Real

FSPEED.FAC

CODE

1 ROUTINE RED.CREATE(A,B,NV)
11 CREATE A TANK
12 READ NAME(TANK), SYS.TYPE(TANK), WPN.TYPE(TANK),
SEC(TANK), PLT(TANK),
CO(TANK),
13 BN(TANK), SQDVH(TANK), PLTLDR(TANK), COCDBR(TANK),
START.AREA(TANK)
14 ALIVE.DEAD(TANK), FIR.SQD(TANK), PIR.SQD(TANK),
BLADE.COND(TANK),
15 CALL FLD.DIST(TANK)
16 LET FLD.INT.DIST(TANK)=RFINF.C
17 LET FSPEED.FAC(TANK)=1.0
18 LET HOLD.FORM(TANK)=999
19 IF BN(TANK) GT RBNS LET RBNS=BN(TANK) ALWAYS
83 RETURN END

6. Routine RES2

Purpose

Routine RES2 allocates space for n-dimensional arrays. This routine is called from the MAIN routine.

Modifications

The following two arrays have been added to the simulation:

6. ROUTINE RED.CREATE(A,B,NV)
1 ROUTINE RED.CREATE(A,B,NV)
11 CREATE A TANK
12 READ NAME(TANK), SYS.TYPE(TANK), WPN.TYPE(TANK),
SEC(TANK), PLT(TANK),
CO(TANK),
13 BN(TANK), SQDVH(TANK), PLTLDR(TANK), COCDBR(TANK),
START.AREA(TANK)
14 ALIVE.DEAD(TANK), FIR.SQD(TANK), PIR.SQD(TANK),
BLADE.COND(TANK),
15 CALL FLD.DIST(TANK)
16 LET FLD.INT.DIST(TANK)=RFINF.C
17 LET FSPEED.FAC(TANK)=1.0
18 LET HOLD.FORM(TANK)=999
19 IF BN(TANK) GT RBNS LET RBNS=BN(TANK) ALWAYS
83 RETURN END

6. Routine RES2

Purpose

Routine RES2 allocates space for n-dimensional arrays. This routine is called from the MAIN routine.

Modifications

The following two arrays have been added to the simulation:
Permanent Attributes Integer

GRUNTLETH (3-D) Integer

This 3-Dimensional array contains the BRL casualty data for anti-personnel mines.

HINLETH (4-D) Integer

This 4-Dimensional array contains the BRL casualty data for anti-tank mines.

CODE

1 ROUTINE RES2
3+ RESERVE HINLETH AS 5 BY 8 BY 2 BY 4
4+ RESERVE GRUNTLETH AS 4 BY 2 BY 35
55 RETURN END

7. Routine RES5

Purpose
Routine RES5 is called from routine MAIN. This routine reads in data.

Modifications
The following arrays have their data read in.

Permanent Attributes Integer

GRUNTLETH (3-D)
HINLETH (4-D)

CODE

1 ROUTINE RES5
2 DEFINE I,J,K,L,M AS INTEGER VARIABLES

174
8. Event TARGET-SELECT

Purpose

Event TARGET-SELECT is called from events DETECT, FIRE, IMPACT, and from itself. This routine selects the highest priority target from an entity's target list for servicing. Selection is based on target range and ammunition availability.

Modifications

This routine has been modified in order to stop a tank from selecting targets and subsequently firing at targets while it is engaged in the following circumstances:

- Pushing a dead plow tank through a minesfield.
- Breaching a gap obstacle with its dozer blade.
- Self breaching a gap obstacle.

The following attributes have been added:

175
Temporary Attribute Integer

BLADE.COND

This unit attribute gives the tank the ability to have a dozer blade characteristic.

0-No blade available.
1-Blade available but not in use.
2-Blade being used or vehicle is self breaching.

PLOW.COND

This unit attribute gives the tank the ability to have a mine plow characteristic.

0-no plow available
1-plow available but not in use
2-plow being used
3-lead vehicle in offset
4-tank that is pushing dead plow tank

TSKED

This attribute serves as a flag to stop further target select.

Brief Explanation

Lines 20-23 Stops the pushing tank from firing.

Lines 24-27 Stops vehicle that is breaching a gap from firing.

CODE

2 UPON TARGET.SELECT(A)
16 IF IPIP(A) = 1
17 RETURN ELSE
18 IF SCHED(A) = 1
19 RETURN ELSE
20 IF PLOW.COND(A) = 4
21 LET TSKED(A) = 1
22 SCHEDULE & TARGET.SELECT(A) IN 15 UNITS
23 RETURN ELSE
24 IF BLADE.COND(A) = 2

176
9. Routine FLD.ACT

Purpose

Routine FLD.ACT is called from the MOVE routine. This routine sorts out what field actions to perform next. (For a more detailed explanation see ref 3)

Modifications

The following routines are now called from routine FLD.ACT:

GAP.DECISION
GAP.ENTRY
GAP.INTERNAL
GAP.LEAVE
MP.DECISION
MP.ENTRY
MP.EXIT
MP.INTERNAL

These routines are discussed in Appendix C. In order for the appropriate routines to be called the following entity attributes have to be used:

Temporary Attribute Integer

FLD.AKT - Code describing the kind of action for pending internal actions.

FLD.NO - Name of the field involved in any pending internal action.
TPNAM.FLD - Field temporary pointer number.

TYP.FLD - Field type code
1 - A mandatory dismount field
2 - A minefield
3 - A minefield decision field
4 - A tank ditch
5 - A road crater
6 - A blown bridge (short-wet gap)
7 - A gap decision field

Brief Explanation

All the additions are self-explanatory in the code. They call one of the appropriate routines mentioned above.

Lines 74-78 This change re-initializes attributes that were changed in routine FLD.ACT. It only takes place upon an entity's departure from a particular type field (one where its FLD.NO was set to the field pointer number).

CODE

2 ROUTINE FLD.ACT(VEH) ""SORTS OUT WHICH FIELD ACTIONS TO PERFORM
5 IF FLD.INT.DIST(VEH) LE 0.0
6 PRINT 1 LINE WITH NAME(VEH), X.CURRENT(VEH), Y.CURRENT(VEH),
7 IAN.FLD(FLD.NO(VEH)), TIME: V AS FOLLOWS
8 FLD INT ACT VEH=##### LOCN = ####### FIELD = ####
9 TIME = ####
8 :: HERE WE CALL ROUTINES TO PERFORM INTERNAL ACTIONS
10 ""
11 IF FLD.AKT(VEH) = 2
12 CALL HP.INTERNAL(VEH)
13 ALWAYS
14 IF FLD.AKT(VEH) = 4 OR FLD.AKT(VEH) = 5 OR
15 FLD.AKT(VEH) = 6
16 CALL GAP.INTERNAL GIVEN VEH
17 ALWAYS

178
18 IF FLD.BDY.DIST(VEH) LE 0.0
19 LET DX = COS.F(DIR.OF.MVT(VEH))
39 IF S1 LE 0.0 AND S1 GE -2.0
40 PRINT 1 LINE WITH NAME(VEH),XCUR,YCUR,
41 TIME.V AS FOLLOWS
FIELD ENTRY VEH=*** LOCN = ******* FIELD = ****
42 "" HERE WE CALL ROUTINES TO PERFORM FIELD ENTRY
BOUNDARY ACTIONS
43 IF COLOR(VEH)=0 AND WPN.TYP(VEH)=8 AND
44 TYP.FLD(FIELD)=1
45 ALWAYS
46 CALL DISMOUNT (VEH,3)
47 IF TYP.FLD (FIELD) = 2
48 CALL NT.ENTRY GIVEN VEH, FIELD
51 IF TYP.FLD (FIELD) = 3
52 CALL NP.DECISION (VEH, FIELD)
54 IF TYP.FLD (FIELD) = 4 OR TYP.FLD(FIELD) = 5 OR
55 TYP.FLD(FIELD) = 6
56 CALL GAP.ENTRY (VEH, FIELD)
57 IF TYP.FLD (FIELD) = 7
58 CALL GAP.DECISION (VEH, FIELD)
60 ALWAYS
61 IF S2 LE 0.0 AND S2 GE -2.0
62 PRINT 1 LINE WITH NAME(VEH),XCUR,YCUR,
63 TIME.V AS FOLLOWS
FIELD EXIT VEH = *** LOCN = ******* FIELD = ****
64 "" HERE WE CALL ROUTINES TO PERFORM FIELD EXIT
BOUNDARY ACTIONS
66 "" IF FLD.AKT(VEH) = 2
67 ALWAYS
68 CALL NT.EXIT GIVEN VEH
69 IF FLD.AKT(VEH) = 4 OR FLD.AKT(VEH) = 5
70 OR FLD.AKT(VEH) = 6
71 ALWAYS
72 "" IF TPNAME.FLD (FIELD), NO.FLD.NO (VEH)
73 LET FLD.INT.DIST (VEH) = BINP.C
74 LET FLD.NO (VEH) = 0
75 LET FLD.AKT(VEH) = 6
76 ALWAYS
78 ALWAYS
79 ALWAYS
80 ENDIF
81 LOOP
82 ALWAYS
83 CALL FLD.DIST (VEH)
84 RETURN
85 END
10. **Routine FLD.CREATE**

**Purpose**

Routine FLD.CREATE creates the field entities. This routine is called from the FLD.INIT routine before the start of the battle.

**Modifications**

Additional field attributes have been added and they needed to be initialized. Several arguments are sent to this routine. Instead of defining them, the attributes that receive their values will be discussed.

**Given Arguments (Real)**

- ANGLE
- P5
- P6
- P7
- P8
- P9
- P10
- P11
- P12
- SAMAJ
- SAMIN

**Global Variable** Integer

- FLD.POINTER (1-D)

Contains the pointer values of all the fields.

**Temporary Attribute** Integer

- TPNAM.FLD - Field temporary pointer number
Temporary Attributes  Real

ANGLE.FLD - Proper angle in radians from east to major axis.

AREA.FLD - Area of this elliptical field

P5.FLD - Field parameter (Appendix D)

P6.FLD - Field parameter (Appendix D)

P7.FLD - Field parameter (Appendix D)

P8.FLD - Field parameter (Appendix D)

P9.FLD - Field parameter (Appendix D)

P10.FLD - Field parameter (Appendix D)

P11.FLD - Field parameter (Appendix D)

P12.FLD - Field parameter (Appendix D)

SAMAJ.FLD - Semi-major axis length

SANIW.FLD - Semi-minor axis length

CODE

1  ROUTINE FLD.CREATE
2  'CREATE ONE FIELD AND SET ITS ATTRIBUTES
3+ GIVEN XC,YC,SAMAJ,SANIN,ANGLE,TYPE,P1,P2,P3,P4,P5,
  P6,P7,P8,P9,P10,
4+ P11,P12 YIELING NAME
5  normally MODE IS REAL
6  DEFINE TYPE, NAME, FIELD AS INTEGER VARIABLES
7  CREATE A FIELD
8  LET XC.FLD(FIELD) = XC
9  LET YC.FLD(FIELD) = YC
10  LET TYP.FLD(FIELD) = TYPE
11+ LET ANGLE.FLD(FIELD) = ANGLE
Routine FLD.INIT

Purpose

Routine FLD.INIT is called once by MAIN before the start of the simulation. It reads the input describing the fields to be created and it calls FLD.CREATE for each such field.

Modifications

The following field attributes have been added to greater enhance the capabilities of an obstacle field. For definitions and input values see Appendix D.
Recursive Variables  Real

p5  
p6  
p7  
p8  
p9  
p10 
p11 
p12  

Global Variables

FLD.POINTER  (1-D)  INTEGER

This 1-Dimensional array contains the pointer values of all the fields. In this routine the space for array is reserved.

TRAF.CONTROL  (3-D)  Real  (Appendix A)

In this routine the space for this 3-Dimensional array is reserved.

Recursive Variables  Integer  (Appendix D)

OB.NUM  

The number of obstacle fields. It is used to dimension the 3-Dimensional array TRAF.CONTROL.

RT.OB.NUM  

The number of routes going through the obstacles. This input variable also dimensions TRAF.CONTROL.

CODE

1  ROUTINE FLD.INIT
2  "CREATE ALL FIELDS THAT ARE IN PLACE AT THE START OF THE BATTLE"
3  NORMALLY NODE IS REAL
12. **Routine BASIC.LOAD**

**Purpose**

This routine is used to initialize the ammunition basic load levels for all elements.

**Modifications**

Subscripted labels for the two new vehicle launched bridges had to be added. These additions only serve as exit labels out of this routine.

**CODE**

```
1 ROUTINE BASIC.LOAD(A)
2 DEFINE A.S.W AS INTEGER VARIABLES
3 LET S = SYS.TYPE(A) LET W = WPN.TYPE(A)
4 LET AMMO1(A) = PJO.CHAR(S, W, 1, 4)
5 LET AMMO2(A) = PJO.CHAR(S, W, 2, 4)
6 LET AMMO3(A) = PJO.CHAR(S, W, 3, 4)
7 LET AMMO4(A) = PJO.CHAR(S, W, 4, 4)
8 GO TO SYSTH(S)
9 'SYSTH(1)' GO TO TANKX(WPN.TYPE(A))
10 'TANKX(1)' LET C.1(A) = 16 LET C.2(A) = 6 LET B.CON(A) = 1
11 GO OUT
12 'TANKX(2)' LET C.1(A) = CAPDS LET C.2(A) = CHEAT
13 LET AP.TOW(A) = CASEHP
14 LET HE.DRAG(A) = CASEHP LET AW1.OB.MSL3(A) = 100
15 LET POH(A) = 1 LET VEH.TYPE(A) = 1 LET B.CON(A) = 1
```
25       RETURN
26       'TANKX(8)'
27       'TANKX(5)'
28       'TANKX(7)'
29       'GO OUT'
30       'OUT'
31       'SYSN(7)'
32       RETURN END
APPENDIX C

ROUTINES AND EVENTS FOR THE ENGINEER MODULE IN STAR

This appendix contains the complete documentation for the Engineer Effects Module for the STAR Combat Model.

1. Routine POP.A.MINE

Purpose

Routine POP.A.MINE determines if an entity is within the casualty producing radius of a mine or if the entity has run over a mine. This routine is called from routine MP.INTERNAL.

Given Arguments

Integer

TNK- Pointer to the element being moved.

Global Variable

Alpha

DAM.ARRAY (1-D)

Contains the possible values for hit.state.

1 NDAM
2 MDAM
3 HDAM
4 FDE
5 DEAD
6 MISS

Global Variable

Integer

DAMAGE.NUM

Indicates the damage status of an entity after having a round impact on or near it.
hit but already mf killed
mobility damage
firepower damage
mobility and firepower damage
catastrophic kill
miss

DEAD.ATKE
Tallies the number of red casualties.

KILLED.MOUNTED
Indicates whether the entity in question was
killed while mounted on his vehicle or while
dismounted.

ONDISK
Indicates whether or not the user desires the
shot list and final attribute list to user
specified disk files.

YES
Indicates a value of 1. It is used in
conditional statements.

Global Variable  Real
TARDIM (3-D)
Contains the target dimensions of all entity
types in the simulation. This is indexed by
SYS.TYPE and WPN.TYPE of the element.

Recursive Variable  Integer
DET.LOC
Is the location of the mine detonation in
relation to the entity. For anti-personnel
Iraq mines this variable contains the range the
entity is from mine detonation. For anti-tank
mines this variable specifies the location of mine detonation on the vehicle.

1 Track hit
2 belly hit

EVEN.BELT

Contains the even mine belt number closest to the actual mine belt to be encountered. This variable is necessary in order to determine how the mines are actually configured on the actual mine belt.

EVEN.N.L.S

Stands for the even number on the left side of the vehicle. This variable helps pin-point the actual mine locations in relation to the entities position on the mine belt.

FLD- Pointer to the field in question.

HALF.BELT

Contains the truncated Integer value when the number of the next encountered mine belt is divided by 2. This is another variable in the sequence used to determine the odd or even numbered representation of belts and mine.

HALF.N.L.S

Contains the truncated integer value that results when the integer number (possible mine location) on the left side of the vehicle is divided by two.

I - Index of a do loop.

J - Subtracts one from the I index.

LEG

Pointer to an element in the platoon of the activating entity (dismounted infantryman).
NUM.LEFT.SIDE

Is the integer location of the left side of the vehicle from the tangent line on the ellipse boundary perpendicular to the semi-major axis.

NUT

Integer mine location under the vehicle/tank.

NXBELT

This local variable receives the value of the entity attribute WHAT.BELT.

WHOCALLED

A flag used by several routines to determine appropriate actions and executions.

Recursive Variable Real

ANGLE

Set to the direction of movement of the entity.

DRAW

Contains the value of a random number from a Uniform(0,1) distribution.

DTOB

Is the distance from the center of a vehicle to the tangent line on the ellipse boundary perpendicular to the semi-major axis.

FKILL

The probability of at least a firepower kill.
EMKILL
The probability of at least a mobility kill.

EMNFKILL
The union of EMKILL and EFKILL.

KAYKILL
The probability of a catastrophic kill.

SIGMA
Contains the value of the difference between the orientation angle of the field ellipse and the direction of movement of the vehicle.

THETA
Receives the value of the field attribute ANGLE.FLD.

VEH.WIDTH
The width of the vehicle in question.

VW
The vehicle width adjusted for SIGMA that crosses the mine belt.

X.PRIME
The X coordinate location of the entity rotated about orientation angle of the field ellipse.

X.MINE
The X coordinate location of the Anti-personnel frag mine.
YMINE

The y coordinate location of the Anti-personnel frag mine.

Routines called

ATRIT

A routine which assesses the attrition against a Target which has been hit. It checks for a catastrophic kill, increments mobility and firepower damage, and checks the FKILL, MKILL, and PFKILL levels.

FIND.A.MINE (this appendix)

PROB.MINE (this appendix)

Sets Used

PLT.UNIT (1-D)

This set is owned by the permanent entity PLATOON LEADER. It contains the list of temporary entities owned by this platoon.

Temporary Attribute Alpha

HIT.STATE

An alpha variable indicating whether or not an element is alive or dead.

Temporary Attribute Integer

ALIVE.DEAD

Indicates whether the entity is alive or dead.

0 alive
1 dead
2 alive mounted in carrier.
This unit attribute gives the tank the ability to have a mine plow characteristic.

COLOR
Indicates the color of the element.
0  red (attacker)
1  blue (defender)

DEFNUM
The current position or activity of an element.
1  full defilade
2  turret defilade
3  firing defilade
4  half vehicle defilade
5  moving (defilade determined by the Terrain model)
6  reached final area in movement

FIRED.AT
Indicates the total number of rounds fired at an entity.

FKILL
Indicates whether an entity has sustained a firepower kill.
0  no
1  yes

FLD.NO
The vehicle attribute that contains the name of the field involved in any pending internal action.

K.HIT
Indicates the number of hits sustained by an entity which were sufficient to cause a catastrophic kill.
KKILL
Indicates whether an entity has sustained a catastrophic kill.
0 no
1 yes

MFKILL
Indicates whether an entity has sustained a simultaneous mobility and firepower kill.
0 no
1 yes

BKILL
Indicates whether an entity has sustained a mobility kill.
0 no
1 yes

NAME
The element number of the entity.

NUM.HIT
Indicates the total number of hits sustained by an entity. This includes no damage hits.

PLT
The number of the platoon to which the entity belongs.

SYS.TYPE
This represents the general class of the system of the entity.
1 Tanks
2 mounted infantry
3 dismounted infantry
4 artillery
5 air
6 air defense
7 bunker

193
8 comm/acq/intel
9 other

WHAT.BELT

This entity attribute keeps the updated belt number that is to be encountered next.

WPN.TYPE

Describes the specific system within the system code. For example, system type 1 is a tank and a weapon type four for this system indicates an AVLB.

Temporary Attribute Real

ANGLE.FLD

Orientation angle in radians measured counterclockwise from east to the major axis of the elliptical field.

DIR.OP.MVT

Indicates the entity's direction of movement measured in radians from east.

P.D

Indicates the accumulated percentage of firepower damage sustained by the entity.

M.D

Indicates the accumulated percentage of mobility damage sustained by the entity.

P3.FLD

Mine Type (1-11)

1 M70 Scatterable Anti-tank mine
2 M56 Scatterable Anti-tank mine
3 M21 Hand-emplaced Anti-tank mine
4 M15 Hand-emplaced Anti-tank mine
5 M19 Hand-emplaced Anti-tank mine
6 M24 Anti-personnel blast mine
7 M25 Anti-personnel blast mine
8 M16 Anti-personnel frag mine
9 ADAM Anti-personnel frag mine
10 M74 Anti-personnel frag mine
11 Claymore Anti-personnel frag mine

P4.FLD (Appendix D)

SAMAJ.FLD

The semi-major axis length of the elliptical field.

SPD

The entity's speed at the end of the most recent movement update.

X.CURRENT

The X-coordinate for the entity as of the last movement update.

XC.FLD

The X-coordinate of the center of an elliptical field.

Y.CURRENT

The Y-coordinate for the entity as of the last movement update.

Z.CURRENT

The elevation for the entity as of the last movement update.
Brief Description

Lines 8-11  Determines if the mine or mine belt was detected and avoided.

Lines 12-42  Subjects dismounted infantry to the effects of anti-personnel mines if applicable.

Lines 14-17  Subjects dismounted infantry to the effects of anti-personnel blast mines.

Lines 18-41  Subjects dismounted infantry to the effects of anti-personnel frag mines.

Lines 43-103  Subjects vehicles to the effects of scatterable and belted/patterned anti-tank mines.

Lines 43-54  Subjects vehicles to the effects of scatterable anti-tank mines. The detonation location under the vehicle is also determined.

Lines 55-102  Subjects the vehicle to the effects of a belt of mines. A vehicle may receive more than one hit at a time. The exact detonation location under a
vehicle is determined because the mines are explicitly played.

**Lines 105-128** Updates the killer-victim score board and it prints out a shot/mine record.

**CODE**

```fortran
ROUTINE POP.A.HINE GIVEN THK
DEFINE THK, DET.LOC, WHOCALED, FLD, NUM.LEFT.SIDE,
HALF.BELT, MD.BELT.EVEN.BELT.HALF. N.L.S, I,J, NOT.LEG AS INTEGER

DEFINE ESKILL, EFKILL, KAYKILL, DRAW THETA,
X.PRIN, TIE, TIE, TIE, TIE, TIE, TIE, TIE, TIE, TIE, TIE, TIE

VARIABLES

LET WHOCALED = 1
LET FLD = FLD.NO(THK)

IF P4.FLD(FLD) GT DETECT AND AVOID
RETURN

OTHERWISE

IF SYS.TYPE(THK) = 3 "DISMOUNTED INFANTRY"
IF P3.FLD(FLD) LE 6.0 "AP-BLAST MINES"
CALL PROB.HINE GIVEN THK, DET.LOC
YIELDING ESKILL, EFKILL, EMMFILL, KAYKILL
CALL ATRIT(THK, TIE, DET.LOC)

ELSE "AP-Frag Mines"
CALL FIND.A.HINE GIVEN THK YIELDING XMINH, XMINH, XMINH
FOR EVERY LEG IN PLT.UNIT(PLT(THK)) WITH
ALIVE, DEAD(LEG) NE 2 AND
SYS.TYPE(LEG) EQ 3, DO

LET DET.LOC =
INT.P(SQRT.P((X.CURRENT(LEG)-XMINH)**2+
(Y.CURRENT(LEG)-YMINH)**2))
IF DET.LOC = 0 ADD 1 TO DET.LOC
ALWAYS
IF P3.FLD(FLD) = 11.0 "Claymore Mine"
IF DET.LOC GT 116 CYCLE
OTHERWISE ADD 24 TO DET.LOC "TO MAKE UP FOR SARAJ AXIS
ELSE IF DET.LOC GT 35 CYCLE
OTHERWISE
CALL PROB.HINE GIVEN THK, DET.LOC
YIELDING ESKILL, EFKILL, EMMFPILL, KAYKILL
CALL ATRIT(THK, TIE, DET.LOC)

197
```
LOOP
  ALWAYS
  ELSE "ROTORIZED VEHICLE
  IF P3.FLD(FLD) LT 6
    IF P3.FLD(FLD) LT 3 " SCATTERABLE MINE FIELD
    LET DRAW=UNIFORM.R(0.,1.)
    IF DRAW LE (.3333)
      LET DET.LOC = 1
    ELSE
      LET DET.LOC = 2
  ALWAYS
  CALL PROB.MINE GIVEN THK, DET.LOC
  YIELDING BPKILL, DPKILL, BNPKILL, EYKILL
  CALL ABST(TNK, THK, BNPKILL, DPKILL, BNPKILL, EYKILL, WROKALLED)
  ELSE
    LET NBELT=WHAT.BELT(TNK)
    LET X.PRIME = X.CURRENT(TNK)*COS.P(THETA)+Y.CURRENT(TNK)*SIN.P(THETA)
    "DTOB =DISTANCE TO THE OUTER BOUNDARY
    LET DTOB = SAMAJ.FLD(FLD)
    LET DIST =DTOB.XC.FLD(FLD)-X.PRIME
    IF DIR.OP.MVMT(TNK) LT 0.0
      LET ANGLE = 360/Radian.C + DIR.OP.MVMT(TNK)
    ELSE
      LET ANGLE = DIR.OP.MVMT(TNK)
    ALWAYS
    LET SIGMA = ANGLE - THETA
    LET VHT.WIDTH = TANK.WIDTH(SYS.TYPE(TNK), WPH.TYPE(TNK), 6)
    LET VN = VHT.WIDTH/COS.P(ABS.P((30/Radian.C)-SIGMA))
    LET NUM.LEFT.SIDE = TRUNC.P(DTOB-VN/2)-1
    LET EVEN.BELT = TRUNC.P(NBELT/2)
    LET HALF.BELT = HALF.BELT+2
    LET HALF.N.L.S = TRUNC.P(NUM.LEFT.SIDE/2)
    LET EVEN.N.L.S =HALF.N.L.S+2
    IF NBELT EQ EVEN.BELT
      IF EVEN.N.L.S EQ NUM.LEFT.SIDE
        SUBTRACT 1 FROM NUM.LEFT.SIDE
        ALWAYS
      ELSE
        IF EVEN.N.L.S NE NUM.LEFT.SIDE
          SUBTRACT 1 FROM NUM.LEFT.SIDE
          ALWAYS
        ALWAYS
        FOR J = 1 TO 10, DO
          LET J = J-1
          LET HUT = J*2.0+NUR.LEFT.SIDE
          IF HUT GT (DTOB-VN/2)
            GO TO 'PRINT'
          OTHERWISE
            IF HUT CYCLE (DTOB-VN/2)
            OTHERWISE
              IF HUT LT (DTOB-VN/3) AND HUT GT (DTOB-VN/3)
              LET DET.LOC = 2
              ELSE
                LET DET.LOC = 1
                ALWAYS
                CALL PROB.MINE GIVEN THK, DET.LOC

198
YIELDING ESKILL,EFSKILL
EENFSKILL,KFSKILL
YIELDING ESKILL,EFSKILL
EENFSKILL,KFSKILL,BROKENDED

CALL ATTRIT(TNK,TNK,EFSKILL,EFSKILL,
EENFSKILL,KFSKILL,BROKENDED)

LOOP

ALWAYS
ALWAYS
ALWAYS
PRINT LET HIT.STATE(TNK) = DAM.ARRAY(DAMAGE.NUM)
START NEW LINE
WRITE NAME(TNK), WPN.TYPE(TNK), TIME.V,
Y.CURRENT(TNK), Y.CURRENT(TNK),
Z.CURRENT(TNK), SPD(TNK), DEP.MON (TNK),
HIT.STATE(TNK) M.D (TNK)
F.D (TNK), SKILL(TNK), ESKILL(TNK),
EFSKILL(TNK), HPKILL(TNK),
SKILL(TNK), K.HIT(TNK), PITCH.At (TNK), NUM.HIT(TNK)
AS 6, "HIT", S 12, 2 1 4, 1 5, S 5, S 22, 3 1 7,
D(8, 1), I 3,
S 5, S 6, S 1, A 4, 2 D(5, 2), 2 1 2, 5 1 3
IF KILLED.HOUNTED = 0
ALWAYS
LET KILLED.HOUNTED = 0
USE UNIT 6 FOR OUTPUT
START NEW LINE
GO TO UP
ELSE IF COLOR(TNK) EQ 0 AND (
DAMAGE.NUM EQ 5 OR
(DAMAGE.NUM EQ 4 OR
DAMAGE.NUM EQ 3 OR DAMAGE.NUM EQ 2) AND
MPKILL(TNK) EQ 1)
AND (MPKILL NUM NE 5 OR MPKILL(TNK) NE 1 )
AND HIT.STATE (TNK) NE "DEAD" AND
HIT.STATE (TNK) NE "ADDED"
ADD 1 TO DEAD.ATKER ALWAYS
IF HIT. STATE (TNK) NE "DEAD" AND
HIT. STATE (TNK) NE "ADDED"
LET HIT.STATE (TNK) = "" ALWAYS
START NEW LINE
RETURN END
2. **Routine FIND.A.MINE**

**Purpose**

Routine FIND.A.MINE finds the X,Y coordinate locations of the Anti-personnel frag mines in question. This routine is called from Routine POP.A.MINE.

**Given Arguments**

- **Integer**
  - **B** - Pointer to the element being moved.

**Yielding Arguments**

- **Real**
  - **XMINE**
    - The X coordinate location of the Anti-personnel frag mine.
  - **YMINE**
    - The Y coordinate location of the Anti-personnel frag mine.

**Recursive Variable**

- **Integer**
  - **FLD**
    - Name of the field involved in this pending action. It serves as a pointer to access other field attributes.

- **Real**
  - **ANGLE**
    - Set to the direction of movement of the entity.
DRAW

Contains the value of a random number from a Uniform(0,1) distribution.

NYN.DIST

The distance in meters that the activating entity is from the Anti-personnel frag mine.

PSI

The direction of movement of the entity minus 90 degrees.

XDELTA

The distance in the X direction from the activating entity's location to the frag mine. The user may think of the NYN.DIST as a vector perpendicular to the direction of movement.

YDELTA

The distance in the Y direction from the activating entity's location to the frag mine. The user may think of the NYN.DIST as a vector perpendicular to the direction of movement.

Temporary Attribute Integer

FLD.NO

The vehicle attribute that contains the name of the field involved in any pending internal action.

Temporary Attribute Real

DIR.OP.MVMT

Indicates the entity's direction of movement measured in radians from the east.
P2. FLD

This field attribute contains the length of the frag mine trip wire in meters.

P3. FLD

Mine Type (1-11)

<table>
<thead>
<tr>
<th></th>
<th>Mine Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>H70 Scatterable Anti-tank mine</td>
</tr>
<tr>
<td>2</td>
<td>H56 Scatterable Anti-tank mine</td>
</tr>
<tr>
<td>3</td>
<td>H21 Hand-emplaced Anti-tank mine</td>
</tr>
<tr>
<td>4</td>
<td>H15 Hand-emplaced Anti-tank mine</td>
</tr>
<tr>
<td>5</td>
<td>H19 Hand-emplaced Anti-tank mine</td>
</tr>
<tr>
<td>6</td>
<td>H14 Anti-personnel blast mine</td>
</tr>
<tr>
<td>7</td>
<td>H25 Anti-personnel blast mine</td>
</tr>
<tr>
<td>8</td>
<td>H16 Anti-personnel frag mine</td>
</tr>
<tr>
<td>9</td>
<td>APAM Anti-personnel frag mine</td>
</tr>
<tr>
<td>10</td>
<td>H74 Anti-personnel frag mine</td>
</tr>
<tr>
<td>11</td>
<td>Claymore Anti-personnel frag mine</td>
</tr>
</tbody>
</table>

I. CURRENT

The X-coordinate for the entity as of the last movement update.

IXC.FLD

The X-coordinate of the center of an elliptical field.

Y. CURRENT

The Y-coordinate for the entity as of the last movement update.

YLC.FLD

The Y-coordinate of the center of an elliptical field.

Brief Explanation
Lines 6-8  Sets the X,Y coordinate location of a Claymore mine to that of the center of the elliptical field.

Lines 10-39 Sets the X,Y coordinate location of the fragmentation mine on a Line that is perpendicular to the entity's direction of movement. The final disposition of the mine in relation to the activating entity is found through a Monte Carlo process.

CODE

1 ROUTINE FIND.A.HINE GIVEN B YIELDING XHINE,YHINE
2 DEFINE B,FLD AS INTEGER VARIABLES
3 DEFINE XHINE,YHINE,DRAW,MYN.DIST,ANGLE,PSI,XDELTA,
4 YDELTA AS REAL VARIABLES
5 LET FLD = FLD.NO(B)
6 IF F3.FLD(FLD) = 1.0 'CLAYMORE MINE
7 LET XHINE = XC.FLD(FLD)
8 LET YHINE = YC.FLD(FLD)
9 ELSE 'OTHER TYPE OF AP,FRAG MINE
10 LET DRAW = UNIFORM.F(0.1, .7)
11 LET MYN.DIST = .707*F2.FLD(FLD)*DRAW
12 LET ANGLE = ABS.F(DIR.OP.MVMT(B))
13 IF DIR.OP.MVMT(B) LT 0.0
14 IF PSI LT 0.0
15 LET PSI = ABS.F(PSI)
16 ELSE
17 LET PSI = (-1.0)*PSI
18 ALWAYS
19 LET XDELTA = MYN.DIST*COS.F(ABS.F(PSI))
20 LET YDELTA = MYN.DIST*SIN.F(ABS.F(PSI))
21 IF DRAW LE .5 'MINE ON RIGHT OF VEHICLE
22 IF PSI LT 0.0
23 LET XHINE = X.CURRENT(B)+XDELTA
24 LET YHINE = Y.CURRENT(B)-YDELTA
25 ELSE
26 LET XHINE = X.CURRENT(B)-XDELTA
27 LET YHINE = Y.CURRENT(B)+YDELTA
28 ALWAYS
29 ELSE 'MINE ON LEFT OF VEHICLE
30 IF PSI LT 0.0
31 LET XHINE = X.CURRENT(B)-XDELTA
32 LET YHINE = Y.CURRENT(B)+YDELTA
33 ELSE
34
36 LET XMINE = X:CURRENT(B) - XDELTA
37 LET YMIN = Y:CURRENT(B) - YDELTA
38 ALWAYS
39 ALWAYS
40 RETURN
41 END
3. **Routine PROB.MINE**

**Purpose**

Routine PROB.MINE accesses the appropriate casualty data from the arrays MINLETH and GRUNLTH. This routine is called from routine POP.A.MINE.

**Given Arguments**

- **Integer**
  - DET.LOC

  *Is the location of the mine detonation in relation to the entity. For Anti-personnel frag mines this variable contains the range the entity is from mine detonation. For Anti-tank mines this variable specifies the location of mine detonation on the vehicle.*

  1. Track hit
  2. Belly hit

- **TNK-** Pointer to the element being moved.

**Yielding Arguments**

- **Real**
  - EFKILL

  *The probability of at least a firepower kill.*

  - EMKILL

  *The probability of at least a mobility kill.*

  - EMNFKILL

  *The union of EMKILL and EFKILL.*

  - KAYKILL

  *The probability of a catastrophic kill.*
Permanent Attribute Integer

GRUNTLETH (3-D)

This 3-dimensional array contains the BRL casualty data for Anti-personnel mines.

MINLETH (4-D)

This 4-dimensional array contains the BRL casualty data for Anti-tank mines.

Recursive Variable Integer

JO

The pointer used to access the appropriate mine casualty data based on vehicle type for Anti-tank mines and firing position for Anti-personnel mines.

HYU.TYPE

This is the pointer used to access Anti-personnel mine casualty data based on appropriate mine type.

TYPE

This variable contains the value of P3.FLD, the mine type in question.

Recursive Variable Real

DRAW

Contains the value of a random number from a Uniform(0,1) distribution.

NAYKILL

Contains the appropriate probability of kill accessed from GRUNTLETH.
Temporary Attribute  Integer

DEPNUM

The current position or activity of an element.

1 full defilade
2 turret defilade
3 firing defilade
4 half vehicle defilade
5 moving (defilade determined by
the terrain model)
6 reached final area in movement

FLD.NO

Name of the field involved in any pending
internal action.

SYS.TYPE

This represents the general class of the system
of the entity.

1 tanks
2 mounted infantry
3 dismounted infantry
4 artillery
5 air
6 air defense
7 bunker
8 comm/ew/acg/intel
9 other

Temporary Attribute  Real

P3.FLD

Mine Type (1-11)

1 M70 Scatterable Anti-tank mine
2 M56 Scatterable Anti-tank mine
3 M27 Handemplaced Anti-tank mine
4 M17 Handemplaced Anti-tank mine
5 M14 Anti-personnel blast mine
6 M25 Anti-personnel blast mine
7 M16 Anti-personnel frag mine
8 APAM Anti-personnel frag mine
9 M74 Anti-personnel frag mine
10 M75 Anti-personnel frag mine
11 Claymore Anti-personnel frag mine

207
Brief Explanation

Lines 6-18 Access the Anti-tank mine BRL data from MINLETH.

Lines 19-51 Access the Anti-personnel mine data from GRUNTLETH. This data is in the form of kill probabilities and must be Monte Carlo-ed against to determine casualties.

CODE

```plaintext
1 ROUTINE PROB.MINE GIVEN TNK, DET.LOC
2   YIELDING EMKILL, EFKILL, ZMNPKILL, KAYKILL
3 DEFINE TNK, TYPE, DET.LOC, JO, SYN.TYPE
4 AS INTEGER VARIABLES
5 DEFINE EMKILL, EFKILL, ZMNPKILL, KAYKILL, HAYKILL,
6 DRAW AS REAL VARIABLES
7 LET TYPE = F3.FLD(FLD.WO(TNK))
8 IF TYPE LT 6.0 'ANTI-TANK mines
9   'WE NEED SOME WAY TO DETERMINE JO-VEHICLE TYPES
10    IF SYS.TYPE(TNK) = 3
11      IF SYS.TYPE(TNK) = 1
12        LET JO = 1
13        ELSE LET JO = 2
14        ALWAYS
15        LET EMKILL = MINLETH(TYPE,JO,DET.LOC,1) /100.
16        LET EFKILL = MINLETH(TYPE,JO,DET.LOC,2) /100.
17        LET ZMNPKILL = MINLETH(TYPE,JO,DET.LOC,3) /100.
18        LET KAYKILL = MINLETH(TYPE,JO,DET.LOC,4) /100.
19    ELSE 'ANTI-PERSONNEL mines
20      IF TYPE LT 8.0 'AP-BLAST
21        LET EMKILL = 1.0
22        LET EFKILL = 1.0
23        LET ZMNPKILL = 1.0
24        LET KAYKILL = 1.0
25      ELSE 'AP-PRAG
26        IF DEFNUM(TNK) = 5 'THE SOLDIER IS STANDING
27          LET JO = 1 'STANDING
28        ELSE 'THE SOLDIER IS IN THE PRONE POSTION
29          LET JO = 2 'PRONE
30          ALWAYS
31        IF TYPE EQ 11 'CLAYMORE MINE
32          LET DET.LOC = INT.P(DET.LOC/10.0)
33        IF DET.LOC = 0
34          ADD 1 TO DET.LOC
35          ALWAYS
36        ALWAYS
37        LET HYN.TYPE = TYPE - 7
```
LET MAYKILL = GRUNTLETH(MYN.TYPE,JO,DET.LOC)/100.0
LET DRAW = UNIFORM.P[0.1,1]
IF MAYKILL GT DRAW 'THE SOLDIER DIES
    LET ESKILL = 1.0
    LET EPKILL = 1.0
    LET EMNKILL = 1.0
ELSE 'THE SOLDIER LIVES TO FIGHT ANOTHER DAY
    LET ESKILL = 0.0
    LET EPKILL = 0.0
    LET EMNKILL = 0.0
ALWAYS
RETURN
END
4. **Routine MF.DECISION**

**Purpose**

Routine MF.DECISION allows the entity and his platoon the options of going to the minefield to attempt a "bull through" or breach, and the option to bypass or lane select, whichever is applicable. This routine is called from routine FLD.ACT.

**Argument**

`A` - Pointer to the element being moved.

`FLD` - Pointer to the field in question.

**Events Scheduled**

- `DIVERT` *(this appendix)*
- `QUIK.MOVE` *(this appendix)*
- `TURN.AROUND` *(this appendix)*

**Global Variable**

`FLD.POINTER` *(1-D) INTEGER*

This 1-dimensional array contains the temporary field pointers. This enables the user to access the obstacles by using the sequential order numbers given the fields on input.

`TRAP.CONTROL` *(3-D) REAL (Appendix A)*

**Recursive Variable**

`C`
COUNTER

Contains the number of five second move increments to get from the decision ellipse through the obstacle under the worst of conditions.

I - Index for a do loop.

KNOW.LEVEL

Contains the value of the field attribute P6.FLD. This is the state of intelligence known about the obstacle.

0  No knowledge exists
1  Knowledge exists
2  Knowledge exists and a lane exists

OBS.FLD

The temporary field pointer of the obstacle.

OBS.NUM

The number of the obstacle to our front. This number is gotten from the field's sequence on input.

STAT.PLOW

This variable gives the platoons status of mine plows.

0  No plow available
1  Plow available but not in use
2  Plow being used
3  Platoon is offsetting
4  Platoon is pushing a dead plow tank

Recursive Variable    Real

DEL.TIME

Receives the plow activation delay time from the minefield attribute P9.FLD.
DIST.AWAY

This is the distance to the obstacle from this decision ellipse.

INTERV

A constant term that is set so that element locations are updated every five seconds during a small portion of the simulation.

PUSH.FACTOR

Receives the value of the minefield attribute P11.FLD, the push dead plow speed degradation factor.

SHAJ.OBS

Receives the value in meters of the semi-major axis of the minefield ellipse to the front.

Routines Called

PLW.ALIVE (this appendix)

Temporary Attribute Integer

BYPASS

This variable is an argument for Event DIVERT. It carries the route number that will cause lateral movement.

DIRC.ON.ET

Indicates whether or not a vehicle is moving forward or backward on his route.

- 0 Vehicle is moving in order of increasing MCP numbers along the route. (forward)
- 1 Vehicle is moving in order of decreasing MCP numbers along the route. (backward)
END.AREA

The number which identifies the ending movement area for this entity.

FLD.AKT

Code describing the kind of action for pending internal actions.

FLD.NO

Name of the field involved in any pending internal action.

FORMACODE

The formation number of the formation to be used by the platoon.

0  Not in formation, vehicle moves along route without offset.

HOLD.FORM

This unit attribute is a storage place, where the unit's formation number for the platoon (FORMACODE) can be placed when lateral route movement is appropriate.

HOME

This variable is an argument for event DIVERT. It carries the MCP number that will be cause lateral movement.

NEXT.MCP

Movement control point number (on the designated route) toward which the element is now moving.

0  end of route has been reached
NUM.POINTER
This variable is an argument for event QUIK.MOVE. It carries the number of the entity to be moved.

REP.NUMBER
This variable is an argument for event QUIK.MOVE. It carries the number of iterations of scheduling left for this event to perform.

ROUTE
Indicates the number of the route along which the element is travelling.
0 Not using a route

SIDE.STEPPE 
This variable is an argument for event DIVERT. It carries the number of the entity to be moved laterally.

START AREA
The number which identifies the starting movement area for this entity.

TWIST
This variable is an argument for event TURN.AROUND. It carries the number of the entity to be turned in the opposite direction.

TYP.FLD
Field Type Code
1 - a mandatory dismount field
2 - a minefield
3 - a minefield decision field
4 - a tank ditch
5 - a road crater
6 - a blown bridge (short-wet gap)
7 - a gap decision field

Temporary Attribute Real
FLD.INT.DIST

This attribute of an entity contains the distance to a pending field internal action.

INT.TIME

This variable is an argument for event QUIK.MOVE. It carries the time interval number when the next event is to be performed.

P1.FLD (Appendix D)
P2.FLD (Appendix D)
P3.FLD (Appendix D)
P4.FLD (Appendix D)
P5.FLD (Appendix D)
P6.FLD (Appendix D)
P9.FLD (Appendix D)
P11.FLD (Appendix D)

SMAAJ.FLD

The semi-major axis length of the elliptical field.

SPD

The entity's speed at the end of the most recent movement update.
XC.FLD

The X-coordinate of the center of an elliptical field.

YC.FLD

The Y-coordinate of the center of an elliptical field.

Brief Explanation

Lines 11-14 Switches the values of HOLD.FORM and FORMACODE. This allows the entity to pick up the route formation if applicable.

Lines 17-20 Checks if this decision ellipse is on a bypass around the minefield and takes appropriate actions if it is.

Lines 21-22 Links to the knowledge attribute, P6.FLD, of the minefield to the front.

Lines 23-24 Turns the entity around if he is coming back from the minefield.

Lines 26-28 Allows the user to stipulate if a bypass or a "bull through" tactic is going to be used.

Lines 31-40 Sets the counter variable to the appropriate number of QUIK.MOVE
iterations needed to get through the minefield.

Line 41 Checks on the status of the plows in the platoon.

Lines 42-46 Determines if the platoon has no knowledge of the minefield or if they have an alive plow, and allows them to proceed toward the minefield.

Lines 47-60 Determines if this minefield has a lane in it. If it does, it allows the entity to move laterally to get to it.

Lines 61-63 Allows the platoon that knows the minefield exists to bypass the minefield.

CODE

```
1 ROUTINE NF DECISION GIVEN A FLD
2 DEFINE A FLD OBS NUM OBS FLD KNOW LEVEL COUNTER
3 STAT FLOW I AS INTEGER VARIABLES
4 DEFINE DEL TIME PUSH FACTOR SHAJ OBS INTERV
5 DIST AWAY AS REAL VARIABLES
6 LET FLD NO (A) = FLD
7 LET FLD AKT (A) = TYP FLD (FLD)
8 LET P4 FLD (FLD) = ROUTE(A)
9 LET P5 FLD (FLD) = NEXT HCP (A)
10 LET INTERV = 5
11 IF HOLD FORM (A) NE 999
12 LET FORMACODE (A) = HOLD FORM (A)
13 LET HOLD FORM (A) = 999
14 ALWAYS
15 "**LINK TO OBSTACLE KNOWLEDGE ATTRIBUTE**
16 LET OBS NUM = INT P3 FLD (FLD)
17 IF OBS NUM = 0
18 "THIS ELLIPSE IS ON A BYPASS ROUTE NO OBSTACLE TO FRONT

217
```
LET FLD.INT.DIST(A) = RINF.C
RETURN

LET OBS.FLD = FLD.POINTER(OBS.NUM)
LET KNOW.LEVEL = INT.F(P6.FLD(OBS.FLD))
IF DIRC.ON. RT(A) = 1
SCHEDULE A TURN.AROUND GIVEN A NOW
ELSE
IF P1.FLD(FLD) NE 0.0 "" BYPASS EXISTS
LET TRAF.CONTROL(OBS.NUM,ROUTE(A),2) = NEXT.MCP(A)
ALWAYS

LET FLD.INT.DIST(A) = RINF.C
IF P6.FLD(FLD) = 0
LET DEL.TIME = P9.FLD(OBS.FLD)
LET PUSH.FACTOR = P11.FLD(OBS.FLD)
LET SMAJ.OBS = SMAJ.FLD(OBS.FLD)
LET DIST.AWAY = SQRT.F((XC.FLD(FLD) - XC.FLD(OBS.FLD))**2 + (YC.FLD(FLD) - YC.FLD(OBS.FLD))**2)
LET COUNTER = INT.F((DEL.TIME + INTERV*(DIST.AWAY/SPD(A))) + ((2*SMAJ.OBS)/(PUSH.FACTOR*SPD(A))))/INTERV
LET P6.FLD(FLD) = COUNTER
ALWAYS
CALL PLOW.ALIVE GIVEN A YIELDING STAT.PLOW
IF KNOW.LEVEL LE 1 OR STAT.PLOW EQ 1
LET COUNTER = INT.F(P6.FLD(FLD))
SCHEDULE A QUIK.MOVE GIVEN A COUNTER INTERV IN INTERV UNITS
RETURN
OTHERWISE
IF KNOW.LEVEL = 3
IF TRAF.CONTROL(OBS.NUM,ROUTE(A),3) NE RINF.C
FOR I = 1 TO DEL.F(TRAFCONTROL(OBS.NUM,*,*)) DO
IF TRAF.CONTROL(OBS.NUM,I,3) EQ RINF.C AND
AND TRAF.CONTROL(OBS.NUM,I,2) NE 0.0 AND
SCHEDULE A DIVERT(A,INT.F(TRAFCONTROL(OBS.NUM,I,1)), INT.F(TRAFCONTROL(OBS.NUM,I,2))) NOW
RETURN
OTHERWISE
LOOP ""FUZZY KNOWLEDGE-LANE EXISTS-NOT PRIVY TO IT
LET KNOW.LEVEL = 2
ALWAYS
ALWAYS
IF KNOW.LEVEL = 2 AND P1.FLD(FLD) NE 0.0
SCHEDULE A DIVERT(A,INT.F(P1.FLD(FLD)), INT.F(P2.FLD(FLD))) NOW
ALWAYS
RETURN
END

218
5. **Routine MF.ENTRY**

**Purpose**

Routine MF.ENTRY allows an entity to take the appropriate actions when encountering a minefield. This routine is called from routine FLD.ACT.

**Given Arguments**

- **A** - Pointer to the entity that has entered the minefield.

- **FLD** - Pointer to the field in question.

**Events Scheduled**

- **QUIK.MOVE** (this appendix)

**Global Variable**

- **TRAF.CONTROL** (3-D) (Appendix A)

**Permanent Attribute**

- **PLT.COND** (1-D)

This attribute of the PLATOON LEADER keeps track of the present action status of the platoon in a minefield. (i.e. plowing, offsetting, pushing, etc.)

**Recursive Variable**

- **BUMP.NUM**

Receives the number of move increments that have been made clearing the appropriate route in question. Its value is stored in the TRAF.CONTROL array.
COUNTER

Contains the number of five second move increments that will take place for this entity. This movement update is accomplished by scheduling event QUICK.MOVE.

INCREMENT

Contains the number of move increments that this entity has to make to keep up with the lead element of its platoon. This number serves as a flag to determine if there is remaining cleared movement to our front or if Routine NINE.SCHED has to be called.

KNOW.LEVEL

Contains the value of the field attribute P6.FLD. This is the state of intelligence known about the obstacle.

- 0: No knowledge exists
- 2: Knowledge exists
- 3: Knowledge exists and a lane exists

LEG

Pointer to an element in the platoon of the activating entity. (dismounted infantryman)

OBS.NUM

Receives the value of the field attribute NAM.FLD. This is the sequential id number in order of field creation.

STAT.PLOW

This variable gives the platoon's status of mine plows.

- 0: No plow available
- 1: Plow available but not in use
- 2: Plow being used
- 3: Platoon is offsetting
- 4: Platoon is pushing a dead plow tank

Recursive Variable: Real

220
CUM.DIST.CLEAR

This is the cumulative distance clear on a route through an obstacle. It receives its value from the TRAP.CONTROL array.

INTERV

A constant term that is set so that element locations are updated every five seconds during a small portion of the simulation.

NF.DET.DIST

This is the distance that a entity will travel into a minefield before it can visually detect that it is in a minefield. (only used for unburied mines)

Routines Called

DROP.PLOW  (this appendix)

MINE.SCHED  (this appendix)

PLW.ALIVE  (this appendix)

WITHDRAW  (this appendix)

Set

PLT.UNIT  (1-D)

This set is owned by the permanent entity PLATOON LEADER. It contains the list of temporary entities owned by this platoon.

Temporary Attribute  Integer
ALIVE.DEAD

Indicates whether the entity is alive or dead.

0 alive
1 dead
2 alive mounted in carrier.

FLD.AKT

Code describing the kind of action for pending internal actions.

FLD.FORM

This unit attribute gives the entity the ability to change formations when encountering an obstacle.

FLD.NO

Name of the field involved in any pending internal action.

NV.STATE

The primary control variable for initiating and stopping movement.

0 In position, do not move.
1 First call to move, do a route select and start to move.
2 Continue movement along a previously selected route.
3 Stop along the route.
4 Next position has been reached, no stop.
5 Final position has been reached, never move again.

MAIN.FLD

This is the sequential id number in order of field creation.

NUM POINTER

This variable is an argument for event QUIK.MOVE. It carries the number of the
entity to be moved.

PLT
The number of the platoon to which the entity belongs.

REP.NUMBER
This variable is an argument for event QUIK.MOVE. It carries the number of iterations of scheduling left for this event to perform.

ROUTE
Indicates the number of the route along which the element is travelling.

0 Not using a route

SYS.TYPE
This represents the general class of the system of the entity.

1 Tanks
2 Mounted infantry
3 Dismounted infantry
4 Artillery
5 Air
6 Air defense
7 Bunker
8 Comm/sw/acg/intel
9 Other

TBUFP.NUM
This unit attribute allows the entity to move in a very detailed manner as a member of his platoon when in an obstacle field.

TYP.FLD
Field Type Code.

1 - a mandatory dismount field
2 - a mine field
3 - a mine field decision field
4 - a tank ditch
5 - a road crater

223
6 - a blown bridge (short-wet gap)
7 - a gap decision field

Temporary Attribute Real

FLD.INT.DIST
This attribute of an entity contains the distance to a pending field internal action.

FSPEED.FAC
This unit attribute gives the entity the ability to slow down because of obstacle encounter.

INT.TIME
This variable is an argument for event QUICK.MOVE. It carries the time interval number when the next event is to be performed.

P1.FLD
The activation level of the minefield.

0 Not activated
1 Activated patterned minefield
# of mines Activated scatterable minefield

P3.FLD (Appendix D)

P6.FLD (Appendix D)

P7.FLD
This minefield attribute contains the lane formation number.

P8.FLD
Lane speed factor through a minefield.
P10.FLD
Plow speed factor while activated in a minefield.

P12.FLD
The minefield detection distance.

SAMAJ.FLD
The semi-major axis length of the elliptical field.

SPD
The entity's speed at the end of the most recent movement update.

Brief Explanation

Lines 18-21 Tests to determine if this minefield is activated yet. If it is not, the entity is allowed free access through the minefield.

Lines 22-30 Updates the entire platoon's location, so that the effects of an Anti-personnel frag mine can be simulated. The distance to a mine encounter is also attained.

Lines 31-35 Checks to see if the bull through is an appropriate tactic and transfers the logic flow to the no knowledge
section of the Routine.

Lines 36-57  The entity takes the actions appropriate when he does not know the minefield exists.

Lines 38-42  If the increment is larger than one, this means that there exists a partial lane. The entity is given this safe distance to travel.

Line 43    A distance to mine encounter is obtained.

Lines 44-54  Determines if the entity is going to hit the next mine or visually detect it first.

Lines 48-52  Keeps the bull through as a separate tactic and does not allow the knowledge of the obstacle to effect the entity's movement.

Lines 58-86  Performs the actions for an entity when he knows the minefield exists.

Lines 59-64  If this route is a lane through the minefield, The formation of the entity is changed to a column, its speed is degraded, and it is allowed
to move through the minefield without
detonating mines.

Line 65  Checks on the status of the platoon's
dlows.

Lines 66-86 Determines if the platoon is going to
breach the minefield or to go back to
the decision ellipse in order to
bypass the obstacle.

Lines 67-78 Allows the entity to go safely over
the partially cleared lane. The
entity's speed is degraded and his
formation is changed to column.

Lines 74-77 The speed degradation depends on the
plowing status of the platoon.

Lines 80-84 This entity's plow is being activated
and the platoon is getting in the
breach formation.

CODE

1 ROUTINE HF.ENTRY GIVEN A,FLD
2 DEFINE A,FLD,KNOW.LEVEL,BURP,NUR,OBV,NUR,STAT.PLOW,
3 INCREMENT,COUNTER,LEG AS INTEGER VARIABLES
4 DEFINE CUM.DIST,CLEAR,FLD.DIST,INTERV
5 AS REAL VARIABLES
6 SUBSTITUTE THESE 6 LINES FOR MATRIX.UPDATE
7 LET TRAP.CONTROL(OBS,NUR,ROUTE(A),E) = ROUTE(A)
8 LET TRAP.CONTROL(OBS,NUR,ROUTE(A),4) =
9 LET TRAP.CONTROL(OBS,NUR,ROUTE(A),5) = BURP,NUR+
10 LET FLD.NO(A) = FLD
11 LET FLD.INT.(A) = TYP.FLD(FLD)
12 LET KNOW.LEVEL = INT.F(P6.FLD(FLD))
13 LET OBS.NUM = NUM.FLD(FLD)
14 LET INTERV = 5.0
15 LET BUMP.NUM = INT.F(TRAF.CTR(0BS.NUM,ROUTE(A),5))
16 LET CUM.DIST.CLR = TRAF.CTR(0BS.NUM,ROUTE(A),3)
17 LET INCRTMNT = BUMP.NUM - TRAF.CTR(0BS.NUM,ROUTE(A),5)
18 IF P1.FLD(FLD) & EQ 0.0 " NOT ACTIVATED YET
19 LET FLD.INT.DIST(A) = RNF.C
20 RETURN
21 OTHERWISE
22 IF SYS.TYPE(A) = 3 AND P3.FLD(FLD) & GE 6.0
23 LET COUNTER = INT.F(SAMAIL.FLD(FLD) & #2.0)/SPD(A)/INTERV) - 2
24 FOR EVERY LEG IN PLT.UNIT(PLT(A) WITH
25 ALIVE.DEAD(LEG) ME 2 AND SYS.TYPE(LEG) EQ 3
26 SCHEDULE A QUIK.MOVE GIVEN LEG, INTERV NOW
27 SCHEDULE A QUIK.MOVE GIVEN A.COUNTER, INTERV IN (2.0*INTERV) UNITS
28 CALL MINIE.SCHD GIVEN A,FLD
29 RETURN
30 OTHERWISE
31 IF TRAF.CTR(0BS.NUM,ROUTE(A),2) = 0.0 OR
32 TRAF.CTR(0BS.NUM,ROUTE(A),2) = 1111.0 OR
33 TRAF.CTR(0BS.NUM,ROUTE(A),2) = 5555.0
34 GO TO 'BULL.THRU'
35 OTHERWISE
36 IF KNOW.LEVEL LE 1 "NO KNOWLEDGE EXISTS
37 "BULL.THRU'
38 IF INCRTMNT GT 0
39 LET FLD.INT.DIST(A) = CUM.DIST.CLR
40 LET BUMP.NUM(A) = BUMP.NUM
41 RETURN
42 OTHERWISE
43 CALL MINIE.SCHD GIVEN A,FLD
44 LET MP.DIST.DIST = P12.FLD(FLD)
45 IF MP.DIST.DIST GT 0
46 IF FLD.INT.DIST(A) GT MP.DIST.
47 LET MP.DIST.DIST(A) = MP.DIST.
48 IF TRAF.CTR(0BS.NUM,ROUTE(A),2) = 0.0
49 LET TRAF.CTR(0BS.NUM,ROUTE(A),2) = 1111.0
50 ELSE
51 LET P6.FLD(FLD) = 1
52 ALWAYS
53 ALWAYS
54 ALWAYS
55 ALWAYS
56 ALWAYS
57 "THEY KNOW THE MINIE FIELD EXISTS
58 IF TRAF.CTR(0BS.NUM,ROUTE(A),3) EQ RNF.C
59 LET FLD.INT.DIST(A) = RNF.C
60 LET FLD.FRM(A) = INT.F(P7.FLD(FLD))
61 LET SPD.F_A(A) = P8.FLD(FLD)
62 RETURN
63 OTHERWISE
64 CALL PLW.ALVIE GIVEN A YIELDING STAT.FLOW
65 IF STAT.FLOW EQ 1
66 IF INCRTMNT GT 0
67 LET FLD.INT.DIST(A) = CUM.DIST.CLR
68 LET BUMP.NUM(A) = BUMP.NUM
69 IF PLT.COND(PLT(A)) = 0
70 LET PLT.COND(PLT(A)) = 1
71 ALWAYS
72 LET FLD.FRM(A) = INT.F(P7.FLD(FLD))
228
IF PLT.COND(PLT(A)) = 2
LET FSPEED.FAC(A) = P10.FLD(FLD)
ELSE
LET FSPEED.FAC(A) = P8.FLD(FLD)
ALWAYS
RETURN
OTHERWISE "DROP/ACTIVATE THE PLOW"
CALL DROP.PLOW GIVEN A,FLD
LET TBUMP.NUM(A) = BUMP.NUM+1
MATRIX.UPDATE
RETURN
OTHERWISE "GO BACKWARDS TOWARD DECISION FIELD"
CALL WITHDRAW GIVEN A
RETURN
END
6. **Routine DROP.PLOW**

**Purpose**

Routine DROP.PLOW simulates the activation of mine plows for the appropriate platoon in a minefield. This routine is called from routines MF.ENTRY and MF.INTRNML.

**Given Arguments**

- **Integer**
  - `B` - Pointer to the entity that is dropping its plow.
  - `FLD` - Pointer to the field in question.

**Events Scheduled**

- **HALT** *(this appendix)*
- **STAND.TO** *(this appendix)*

**Permanent Attribute**

- **Integer**
  - `PLT.COND (1-D)`

*This attribute of the PLATOON LEADER keeps track of the present action status of the platoon in a minefield. (i.e. plowing, offsetting, pushing, etc.)*

**Recursive Variable**

- **Integer**
  - `TYK`

*Pointer to an element in the platoon of the activating entity.*

**Routines Called**

- **MINE.SCHED** *(this appendix)*

230
Set

PLT.UNIT (1-D)

This set is owned by the permanent entity platoon leader. It contains the list of temporary entities owned by this platoon.

Temporary Attribute Integer

ALIVE.DEAD

Indicates whether the entity is alive or dead.

0 Alive
1 Dead
2 Alive mounted in carrier.

BAGGED.BOG

This variable is an argument for event STAND.TO. It carries the number of the entity to be moved.

FLD.FORM

This unit attribute gives the entity the ability to change formations when encountering an obstacle.

FLD.NO

Name of the field involved in any pending internal action.

PLT

The number of the platoon to which the entity belongs.

SLOW.DOWN

This variable is an argument for event HALT. It carries the number of the entity to be stopped.
Temporary Attribute  Real

FSPEED.FAC

This unit attribute gives the entity the ability to slow down because of obstacle encounter.

P7.FLD

This minefield attribute contains the lane formation number.

P9.FLD

This minefield attribute contains the plow activation delay time.

P10.FLD

This minefield attribute contains the plow speed factor.

Brief Explanation

Lines 3-13  Stops the platoon for a certain period of time, degrades their speed in order to plow, and schedules a start up some time in the future.

Lines 7-9  Only degrades the speed for vehicles in the minefield.

Lines 11-13  Changes the activating entity's formation to a breach column and obtains a distance to next mine encounter.
CODE

1 ROUTINE DROP.FLOW GIVEN B, FLD
2 DEFINE B, FLD, TNK AS INTEGER VARIABLES
3 FOR EVERY TNK IN PLT.UNIT(PLT(B))
4 WITH ALIVE.DEAD(TNK) NE 2, DO
5 SCHEDULE A HALT GIVEN TNK NOW
6 SCHEDULE A STAND.TO GIVEN TNK IN
7 IF FLD.NO(TNK) EQ FLD
8 LET FSPEED.PAC(TNK) = P10.FLD(FLD)
9 ALWAYS
10 LOOP
11 LET PLT.COND(PLT(B)) = 2
12 LET FLD.FORM(B) = INT.F(P7.FLD(FLD))
13 CALL MINE.SCHED GIVEN B, FLD
14 RETURN
15 END
7. **Routine WITHDRAW**

**Purpose**

Routine WITHDRAW stops, turns around, and starts the platoon up again. This routine is called from routines MF.ENTRY and MF.INTERNAL.

**Given Arguments**

- Integer
  - B - Pointer to the entity that is withdrawing.

**Events Scheduled**

- HALT (this appendix)
- STAND.TO (this appendix)
- TURN.AROUND (this appendix)

**Recursive Variable**

- Integer
  - TIK

     Pointer to an element in the platoon of the activating entity.

**Set**

- PLT.UNIT (1-D)

This set is owned by the permanent entity platoon leader. It contains the list of temporary entities owned by this platoon.

**Temporary Attribute**

- Integer
  - ALIVE.DEAD

Indicates whether the entity is alive or dead.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alive</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dead
Alive mounted in carrier.

BAGGED.BOX

This variable is an argument for event STAND.TO. It carries the number of the entity to be moved.

DIRC.ON. RT

Indicates whether or not a vehicle is moving forward or backward on his route.

- 0 Vehicle is moving in order of increasing MCP numbers along the route. (forward)
- 1 Vehicle is moving in order of decreasing MCP numbers along the route. (backward)

END AREA

The number which identifies the ending movement area for this entity.

PLT

The number of the platoon to which the entity belongs.

SLOW.DOWN

This variable is an argument for event HALT. It carries the number of the entity to be stopped.

START AREA

The number which identifies the starting movement area for this entity.

TWIST

This variable is an argument for event TURN.AROUND. It carries the number of the entity to be turned in the opposite direction.
Temporary Attribute Real

FLD.INT.DIST

This attribute of an entity contains the distance to a pending field internal action.

Brief Explanation

Lines 3-11 Turns around the vehicles in the platoon that are moving forward.

Lines 6-8 Stops, turns around, and starts the appropriate entity in the next five seconds.

CODE

1 ROUTINE WITHDRAW GIVEN B
2 DEFINE B,TNK AS INTEGER VARIABLES
3 IF DIREC.ON.BT(B) EQ 0
4 FOR EVERY TNK IN PLT.UNIT(PLT(B)) WITH ALIVE.DEAD(TNK) NE 2, DO
5 SCHEDULE A HALT GIVEN TNK NOW
6 SCHEDULE A TURN.AROUND GIVEN TNK IN 4.5 UNITS
7 SCHEDULE A STAND.TO GIVEN TNK IN 5 UNITS
8 LET FLD.INT.DIST(TNK) = RINF.C
9 LOOP
10 ALWAYS
11 RETURN
12 END

236
8. **Routine MF-INTERNAL**

**Purpose**

Routine MF-INTERNAL allows an entity to take the appropriate internal actions in a minefield. This routine is called from routine FLD.ACT.

**Given Arguments**

*Integer

A

Pointer to the entity that activated the minefield internal action.

**Global Variable**

*Real

TRAP.CONTROL (3-D) (Appendix A)

**Permanent Attribute**

*Integer

PLT.COND (1-D)

This attribute of the PLATOON LEADER keeps track of the present action status of the platoon in a minefield. (i.e. plowing, offsetting, pushing, etc.)

**Recursive Variable**

*Integer

BUMP.NUM

Receives the number of move increments that have been made clearing the appropriate route in question. Its value is stored in the TRAP.CONTROL array.

**FLD-** Pointer to the field in question.
INCREMENT

Contains the number of move increments that this entity has to make to keep up with the lead element of his platoon. This number serves as a flag to determine if there is remaining cleared movement to our front or if routine SCHED has to be called.

KNOW.LEVEL

Contains the value of the field attribute P6.FLD. This is the state of intelligence known about the obstacle.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No knowledge exists</td>
</tr>
<tr>
<td>2</td>
<td>Knowledge exists</td>
</tr>
<tr>
<td>3</td>
<td>Knowledge exists and a lane exists</td>
</tr>
</tbody>
</table>

OBS.NUM

Receives the value of the field attribute NAM.FLD. This is the sequential id number in order of field creation.

STAT.PLOW

This variable gives the platoon's status of mine plows.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No plow available</td>
</tr>
<tr>
<td>1</td>
<td>1-plow available but not in use</td>
</tr>
<tr>
<td>2</td>
<td>2-plow being used</td>
</tr>
<tr>
<td>3</td>
<td>3-platoon is offsetting</td>
</tr>
<tr>
<td>4</td>
<td>4-platoon is pushing a dead plow tank</td>
</tr>
</tbody>
</table>

TMP.FORM

Contains the highest formation number attained by the entity's platoon.

TNK

Pointer to an element in the platoon of the activating entity.

Recursive Variable Real

238
CUM.DIST.CLEAR

This is the cumulative distance clear on a route through an obstacle. It receives its value from the TRAP.CONTROL array.

DRAW

Contains the value of a random number from a Uniform(0,1) distribution.

MOV.DIST

The distance that routine MOMENTUM returns. This is the distance to the lead tank in the platoon.

Routines Called

DROP.PLOW (this appendix)

HIDER

A routine used to determine the micro-terrain elevation for a selected element.

MINE.SCHED (this appendix)

MOMENTUM (this appendix)

PLW.ALIVE (this appendix)

POP.A.MINE (this appendix)

WITHDRAW (this appendix)
PLT.UNIT  (I-D)

This set is owned by the permanent entity PLATOON LEADER. It contains the list of temporary entities owned by this platoon.

Temporary Attribute       Integer

DEPMUM

The current position or activity of an element.

1  full defilade
2  turret defilade
3  firing defilade
4  half vehicle defilade
5  moving (defilade determined by the terrain model)
6  reached final area in movement

FLD.FORM

This unit attribute gives the entity the ability to change formations when encountering an obstacle.

FLD.NO

Name of the field involved in any pending internal action.

SKILL

Indicates whether an entity has sustained a catastrophic kill.

<table>
<thead>
<tr>
<th>0</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
</tr>
</tbody>
</table>

SKILL

Indicates whether an entity has sustained a mobility kill.

<table>
<thead>
<tr>
<th>0</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
</tr>
</tbody>
</table>
**MV. STATE**

The primary control variable for initiating and stopping movement.

0  In position, do not move.
1  First call to move, do a route select and start to move.
2  Continue movement along a previously selected route.
3  Stop along the route.
4  Next position has been reached, so stop.
5  Final position has been reached, never move again.

**NAH.FLD**

This is the sequential id number in order of field creation.

**PLOW.COMD**

This unit attribute gives the tank the ability to have a mine plow characteristic.

0-no plow available
1-plow available but not in use
2-plow being used
3-lead vehicle in offset
4-tank that is pushing dead plow tank

**PLT**

The number of the platoon to which the entity belongs.

**ROUTE**

Indicates the number of the route along which the element is travelling.

0  Not using a route

**SYS.TYPE**

This represents the general class of the system of the entity.

1  tanks
2  mounted infantry
3  dismounted infantry
TBUMP.NUM

This unit attribute allows the entity to move in a very detailed manner as a member of his platoon when in an obstacle field.

Temporary Attribute Real

FLD.INT.DIST

This attribute of an entity contains the distance to a pending field internal action.

FSPEED.FAC

This unit attribute gives the entity the ability to slow down because of an obstacle encounter.

P3.FLD

Mine Type (1-11)

1  M70 Scatterable Anti-tank mine
2  M56 Scatterable Anti-tank mine
3  M21 Hand-emplaced Anti-tank mine
4  M15 Hand-emplaced Anti-tank mine
5  M19 Hand-emplaced Anti-tank mine
6  M14 Anti-personnel blast mine
7  M25 Anti-personnel blast mine
8  M16 Anti-personnel frag mine
9  ADAM Anti-personnel frag mine
10  M74 Anti-personnel frag mine
11  Claymore Anti-personnel frag mine

P5.FLD

Probability of pushing a dead plow tank.
(1-P5.FLD) = offset probability
**P6.FLD**

Knowledge level of this minefield.

- **0**: No knowledge exists.
- **1**: No knowledge exists, the entity will see the minefield before a mine goes off.
- **2**: Knowledge exists.
- **3**: Knowledge exists and a lane exists.

**P7.FLD**

This minefield attribute contains the lane formation number.

**P11.FLD**

Pushing the dead plow speed factor.

**Brief Explanation**

**Lines 15-20**

The dismounted infantry entity detonates a mine and if he survives, a new distance to the next mine detonation is obtained.

**Lines 22-30**

Changes the knowledge flag so that the "bull through" option will be played.

**Lines 31-48**

Simulates the no knowledge exists state of this minefield.

**Lines 32-34**

Detonates a mine for the unaware entity.
Lines 35-37 Only changes the knowledge state when it needs to be changed. This also serves as a check to make sure the "bull through" entity does not change the knowledge state.

Lines 39-47 If the platoon has a plow it will activate it, otherwise the platoon will turn around and go back to the decision ellipse to bypass around this minefield.

Lines 49-137 Performs the appropriate internal actions when the minefield is known to exist.

Lines 50-53 If there is now a lane on this route, the rest of the platoon elements move free, undamaged.

Lines 54-69 If there is any cleared distance in front of an entity, this distance is travelled first.

Line 71 Checks the status of the platoon plows.
Performs certain actions when a platoon has an alive plow.

If the platoon is already plowing, this lead plow tank will receive a distance to the next minefield internal action (mine encounter), and will continue to plow.

If the platoon has not been plowing but has a plow or plows alive, they will now be activated.

The platoon had an alive plow on entry of the minefield, but it is now dead. The platoon will now withdraw and take a bypass around the this obstacle.

If the "bull through" tactic is appropriate, the platoon will begin on an offset sequence.

If the platoon has offset or pushed the dead plow tank before and the pusher tank is dead (if applicable), then an offset sequence is started or continued.
Lines 95-134 If the platoon had been plowing and the plow tanks are dead, the platoon has the option to either offset around the dead plow or push the dead plow tank.

Lines 97-102 This entity will push the dead plow tank.

Lines 103-108 Changes the formation of the entire platoon in this minefield for a first time offset.

Lines 109-114 Upgrades the speed of the entity to the maximum allowed by the terrain and it reassures that this entity is the lead element of his platoon. A mine detonation also occurs.

Lines 115-132 Offsets the rest of the platoon elements if the activating entity is at least a mobility kill.

Lines 129-131 If this is a patterned minefield and a tank has died in this mine belt, the other tanks in the activating entity's platoon must still go through this same belt.
Lines 135-137 Obtain a new mine encounter distance and update the TRAP.CONTROL array.

CODE

1 ROUTINE MF.INTERNAL GIVEN A
2 DEFINE A FLD.KNO.L,LEVEL, BUMP.NUM, OBS.NUM, STAT.PLOW.
3 INCREMENT, LMK, TRAP.FORM AS INTEGER VARIABLES
4 DEFINE CUM.DIST.CLEAR, DRAW, MOV.DIST
5 AS REAL VARIABLES
6 SUBSTITUTE THESE 3 LINES FOR MATRIX.UPDATE
7 LET TRAP.CONTROL(OBS.NUM, ROUTE(A), 3) = CUM.DIST.CLEAR + FLD.INT.DIST(A)
8 LET TRAP.CONTROL(OBS.NUM, ROUTE(A), 4) = FLD.INT.DIST(A)
9 LET FLD = FLD.NO(A)
10 LET KNO.L.LEVEL = INT.F(P6.FLD(FLD))
11 LET OBS.NUM = HAM.FLD(FLD)
12 LET BUMP.NUM = INT.F(TRAP.CONTROL(OBS.NUM, ROUTE(A), 5))
13 LET CUM.DIST.CLEAR = TRAP.CONTROL(OBS.NUM, ROUTE(A), 3)
14 LET INCREMENT = BUMP.NUM - TBUMP.NUM(A)
15 IF SYS.TYPE(A) = 3 "DISMOUNTED INFANTRY"
16 CALL POP.A.MINR GIVEN A
17 IF KILL(A) NE 1 "A SOLDIER HIT IS DEAD"
18 CALL HINE.SCHD GIVEN A, FLD
19 ALWAYS RETURN
20 OTHERWISE
21 IF TRAP.CONTROL(OBS.NUM, ROUTE(A), 2) = 0.0 OR
22 IF TRAP.CONTROL(OBS.NUM, ROUTE(A), 2) = 1111.0-
23 IF TRAP.CONTROL(OBS.NUM, ROUTE(A), 2) = 0.0
24 LET KNO.L.LEVEL = 0
25 ELSE
26 LET KNO.L.LEVEL = 1
27 ALWAYS RETURN
28 OTHERWISE
29 ALWAYS
30 ALWAYS
31 IF KNO.L.LEVEL LE 1 "NO KNOWLEDGE EXISTS"
32 IF KNO.L.LEVEL EQ 0
33 CALL POP.A.HINE(A)
34 ALWAYS
35 IF P6.FLD(FLD) LT 2 "KNOWLEDGE EXISTS"
36 LET P6.FLD(FLD) = 2 "KNOWLEDGE EXISTS"
37 ALWAYS
38 IF TRAP.CONTROL(OBS.NUM, ROUTE(A), 2) NE 5555.0
39 CALL PLOW.ALIVE GIVEN A, YIELDING STAT.PLOW
40 IF STAT.PLOW EQ 1 "DROP PLOW OR GO BACK"
41 CALL DROP.PLOW GIVEN A, FLD
42 LET TRAMP.NUM(A) = BUMP.NUM+1
43 MATRIX.UPDATE
44 RETURN
45 OTHERWISE "GO BACKWARDS TOWARD DECISION FIELD"
46 CALL WITHDRAW GIVEN A
47 RETURN
48 OTHERWISE "BULL-THRU"
49 ALWAYS "KNOWLEDGE EXISTS ABOUT THE MINEFIELD"
50 IF TRAP.CONTROL(OBS.NUM, ROUTE(A), 3) EQ HINE.C
51 LET FLD.INT.DIST(A) = TRAP.CONTROL(OBS.NUM, ROUTE(A), 3)
52 ALWAYS RETURN
53
IF INCREMENT GT 0  
IF INCREMENT EQ 1
LET FLD.INT.DIST(A) = TRAP.CONTROL(OBS.NUM, ROUTE(A), 4)
ELSE
CALL MOMENTUM GIVEN A, BUMP.NUM YIELDING MOV.DIST
LET FLD.INT.DIST(A) = MOV.DIST
ALWAYS
LET TBUMP.NUM(A) = BUMP.NUM
FOR EVERY TNK IN PLT.UNIT(PLT(A)) WITH SYS.TYPE(TNK) NE 3, DO
IF FLD.FORM(TNK) GT TMP.FORM
LET TMP.FORM = FLD.FORM(TNK)
ALWAYS
LOOP
LET FLD.FORN(TNK) = TMP.FORM
RETURN
OTHERWISE
CALL PLW.ALIVE GIVEN A YIELDING STAT.PLW
IF STAT.PLW EQ 1
IF PLT.COND(PLT(A)) EQ 2
CALL MINE.SCUD GIVEN A, FLD
LET TBUMP.NUM(A) = BUMP.NUM + 1
ELSE "TO DROP PLLOW"
CALL DROP.GIVEN A, FLD
LET TBUMP.NUM(A) = BUMP.NUM + 1
ALWAYS
MATRIX.UPDATE
RETURN
OTHERWISE "THE PLOWS ARE DEAD"
IF PLT.COND(PLT(A)) EQ 1
"GO BACKWARDS TOWARD DECISION FIELD"
CALL WITHDRAW GIVEN A
RETURN
OTHERWISE
IF PLT.COND(PLT(A)) EQ 0 "BULL-THRU CASE"
GO TO "OFFSET"
OTHERWISE
IF PLT.COND(PLT(A)) GE 3
"PUSHED OR OFFSET BEFORE"
IF STAT.PLW LE 3 "PUSHER OR LEAD TANK DEAD"
GO TO "OFFSET"
OTHERWISE "PUSHER OR LEAD TANK STILL ALIVE"
IF PLT.COND(PLT(A)) EQ 2 "HAD BEEN PLOWING BUT PLOW DIED"
LET DRAW = UNIFORM.F(0., 1., 7)
IF DRAW LE P5.FLD(FLD)
"PUSH THE DEAD PLOW TANK"
LET PSPEED.FAC(A) = P1.FLD(FLD)
LET PLOW.COND(A) = 4
LET DEFWHM(A) = 4",1/2" CALL HIDER(A)
LET PLOW.COND (PLT(A)) = 4
ELSE "OFFSET AROUND DEAD PLOW TANK"
FOR EVERY TNK IN PLT.UNIT(PLT(A)) WITH SYS.TYPE(TNK) NE 3, DO
IF FLD.NO(TNK) EQ FLD
LET FLD.FORM(TNK) = INT.F(P7.FLD(FLD))+1
ALWAYS
LOOP
LET PSPEED.FAC(A) = 1.0
LET PLOW.COND(A) = 3
IF PLT.COND(PLT(A)) NE 0 "NOT BULL-THRU CASE"
CALL POP.A.HINE(A)
ALWAYS
LET PLT_COND(PLT(A)) = 3
IF KILL(A) = 1 OR KILL(A) = 1
FOR EVERY TNK IN PLT_UNIT(PLT(A))
WITH SYS_TYPE(TNK) BE 3, DO
IF KILL(TNK) NE 1 AND KILL(TNK) NE 1
IF FLD_NO(TNK) EQ FLD
IF FLD_FORM(TNK) LT
LET F(TNK, FLD(FLD))
**Bull-Thru Offset
LET FLD_FORM(TNK) =
INT(F(TNK, FLD(FLD)) + 1
ELSE**Regular Offset
ADD 1 TO FLD_FORM(TNK)
ALWAYS
ALWAYS
ALWAYS
ALWAYS
ALWAYS
ALWAYS
ALWAYS
CALL MINE_SCHED GIVEN A, FLD
MATRIX_UPDATE
LET TBUMP_NUM(A) = BUMP_NUM + 1
RETURN
END
9. **Routine MF.EXIT**

**Purpose**

Routine MF.EXIT performs the minefield exit actions for the activating/moving entity. This routine is called from routine FLD.ACT.

**Given Arguments**

- **Integer**
  - A
    - Pointer to the entity that activated the minefield internal action.

**Global Variable**

- **Real**
  - TRAP.CONTROL (3-D) (Appendix A)

**Permanent Attribute**

- **Integer**
  - PLT.COMD (1-D)
    - This attribute of the PLATOON LEADER keeps track of the present action status of the platoon in a minefield. (i.e. plowing, offsetting, pushing, etc.)

**Recursive Variable**

- **Integer**
  - FLD-
    - Pointer to the field in question.

**OBS.NUM**

- Receives the value of the field attribute NAB.FLD. This is the sequential id number in order of field creation.

**Routines Called**

- 250
HIDER

A Routine used to determine the micro-terrain elevation for a selected element.

Temporary Attribute Integer

DEPNUM

The current position or activity of an element.

1 full defilade
2 turret defilade
3 firing defilade
4 half vehicle defilade
5 moving (defilade determined by the terrain model)
6 Reached final area in movement

DIRC.ON. BT

Indicates whether or not a vehicle is moving forward or backward on his route.

0 Vehicle is moving in order of increasing MCP numbers along the route. (forward)
1 Vehicle is moving in order of decreasing MCP numbers along the route. (backward)

FLD.FORM

This unit attribute gives the entity the ability to change formations when encountering an obstacle.

FLD.NO

Name of the field involved in any pending internal action.

FAH.FLD

This is the sequential id number in order of field creation.
PLOW.COND

This unit attribute gives the tank the ability to have a mine plow characteristic.

0 - no plow available
1 - plow available but not in use
2 - plow being used
3 - lead vehicle in offset
4 - tank that is pushing dead plow tank

PLT

The number of the platoon to which the entity belongs.

ROUTE

Indicates the number of the route along which the element is travelling.

0 - Not using a route

SYS.TYPE

This represents the general class of the system of the entity.

1 - tanks
2 - mounted infantry
3 - dismounted infantry
4 - artillery
5 - air
6 - air defense
7 - bunker
8 - com/ew/acq/intel
9 - other

TBUMP.NUM

This unit attribute allows the entity to move in a very detailed manner as a member of his platoon when in an obstacle field.

Temporary Attribute  Real

FSPEED.FAC

This unit attribute gives the entity the ability to slow down because of obstacle encounter.
P1.FLD

The activation level of the minefield.

<table>
<thead>
<tr>
<th>0</th>
<th>Not activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Activated patterned minefield</td>
</tr>
</tbody>
</table>

# of mines

P6.FLD

Knowledge level of this minefield.

| 0 | no knowledge exists |
| 2 | knowledge exists |
| 3 | knowledge exists and a lane exists |

Brief Explanation

Lines 3-5  If the entity is dismounted infantry, do nothing.

Lines 9-18 Updates the TRAP.CONTROL array for this cleared lane and lets others know that a lane exists on this route through the minefield.

Lines 10-12 Does not let the knowledge level change if the entity has not hit or seen a mine.

Lines 13-15 Prevents TRAP.CONTROL array changes when the minefield is not activated yet.
Lines 16-19 If the entity was the push tank or the lead element, then this status of the entity no longer holds.

Lines 21-23 Re-initializes changed entity attributes.

CODE

1 ROUTINE MF.EXIT GIVEN A
2 DEFINE A. FLD, OBS. NUM AS INTEGER VARIABLES
3 IF SYS. TYPE(A) = 3 "DISMOUNTED INFANTRY"
4 RETURN
5 OTHERWISE
6 LET FLD = FLD.WO(A)
7 LET OBS. NUM = BN. FLD.FLD(FLD)
8 LET ESPEED.FAC(A) = 1.0
9 IF DIREC. ON. BT(A) = 0
10 IF TRUPE. NUM(A) GT 1
11 LET P6. FLD(FLD) = 3
12 ALWAYS
13 IF F1. FLD(FLD) NE 0.0 'ACTIVE MINEFIELD'
14 LET TRAP. CONTROL(OBS. NUM, ROUTE(A), 3) = RINF.C
15 ALWAYS
16 IF PLOW. COND(A) GT 1
17 LET DEFNUM(A) = 5 'MOVING' CALL HIDER(A)
18 ALWAYS
19 LET PLOW. COND(A) = 0
20 ALWAYS
21 LET FLD. FORM(A) = 0
22 LET TBUMP. NUM(A) = 0
23 LET PLT. COND(PLT(A)) = 0
24 RETURN
25 END
10. **Event HALT**

**Purpose**

Event HALT stops the moving entity. This event is important because it allows the user to get out of the PLD.ACT and MOVE routines. This event is scheduled from routines DROP.PLOW, WITHDRAW, GAP.ENTRY, and GAP.INTERNAL.

**Given Arguments**

- **Integer**
  - B - The pointer to the entity that has activated this stop event.

**Temporary Attribute**

- **Integer**
  - MV.STATE

  The primary control variable for initiating and stopping movement.

  0  In position, do not move.
  1  First call to move, do a route select and start to move.
  2  Continue movement along a previously selected route.
  3  Stop along the route.
  4  Next position has been reached, so stop.
  5  Final position has been reached, never move again.

**SLOW.DOWN**

This variable is an argument for event HALT. It carries the number of the entity to be stopped.

**Brief Explanation**

255
Lines 1-5  Changes the MV.STATE of the entity so that it will slow down and stop moving.

CODE

1 EVENT HALT GIVEN B
2 DEFINE B AS AN INTEGER VARIABLE
3 LET MV.STATE(B) = 3
4 RETURN
5 END
11. **Event TURN.AROUND**

**Purpose**

Event TURN.AROUND turns the entity around in the opposite direction of present travel. This event is scheduled from Routines MP.DECISION, WITHDRAW, GAP.DECISION, GAP.ENTRY, and from event HEAVY.JUNK.

**Given Arguments**

Integer

B - The pointer to the entity that is going to turn around.

**Recursive Variable**

Integer

THP.AREA

This variable stores the value of the STATE.AREA, so that the values of START.AREA and END.AREA can be switched.

**Temporary Attribute**

Integer

END.AREA

The number which identifies the ending movement area for this entity.

**SKILL**

Indicates whether an entity has sustained a catastrophic kill.

<table>
<thead>
<tr>
<th>0</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**SKILL**

Indicates whether an entity has sustained a mobility kill.

<table>
<thead>
<tr>
<th>0</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
</tr>
</tbody>
</table>
START. AREA

The number which identifies the starting movement area for this entity.

TWIST

This variable is an argument for event TURN.AROUND. It carries the number of the entity to be turned in the opposite direction.

Brief Explanation

Lines 3-7 Switches the START.AREA and END.AREA of the alive entity in question.

CODE

1 EVENT TURN.AROUND GIVEN B
2 DEFINE B, TMP. AREA AS INTEGER VARIABLES
3 IF SKILL(B) NE 1 AND MKILL(B) NE 1
4 LET TMP. AREA = START. AREA(B)
5 LET START. AREA(B) = END. AREA(B)
6 LET END. AREA(B) = TMP. AREA
7 ALWAYS
8 RETURN
9 END
12. **Event STAND.TO**

**Purpose**

Event STAND.TO starts the activating entity moving again. This event is scheduled from routines DROP.PLOW, WITHDRAW, and GAP.INTERNAL. It is also scheduled from events GAP.BREACH, HEAVY.JUNK, and GAP.JOCK.

**Given Argument**  
**Integer**

B - The pointer to the entity that is going to start moving again.

**Temporary Attribute**  
**Integer**

BAGGED.BOY

This variable is an argument for Event STAND.TO. It carries the number of the entity to be moved.

**KILL**

Indicates whether an entity has sustained a catastrophic kill.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>yes</td>
</tr>
</tbody>
</table>

**MKILL**

Indicates whether an entity has sustained a mobility kill.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>yes</td>
</tr>
</tbody>
</table>

**NV.STATE**

The primary control variable for initiating and stopping movement.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>In position, do not move.</td>
</tr>
</tbody>
</table>

259
First call to move, do a route select and start to move.
Continue movement along a previously selected route.
Stop along the route.
Next position has been reached, so stop.
Final position has been reached, never move again.

Temporary Attribute Real

SPD

The entity's speed at the end of the most recent movement update.

T.SPD

The simulation time at which the most recent movement update ended. The time that SPD was last set.

Brief Explanation

Lines 3-7 Starts the entity moving again now from a standstill.

CODE

1 EVENT STAND.TO GIVEN B
2 DEFINE B AS AN INTEGER VARIABLE
3 IF SKILL(B) NE 1 AND MKILL(B) NE 1
4 LET T.SPD(B) = TIME.V
5 LET BV.STATE(B) = 2
6 LET SPD(B) = 0.0
7 ALWAYS
8 RETURN
9 END
13. **Event DIVERT**

**Purpose**

Event DIVERT allows/triggers lateral movement for an entity going from one decision ellipse to another. This event is scheduled from routines NF.DECISION and GAP.DECISION.

**Given Arguments**

<table>
<thead>
<tr>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

- **3** - The pointer to the entity that is going to start lateral movement.

<table>
<thead>
<tr>
<th>MCP</th>
</tr>
</thead>
</table>

- **MCP**
  - This argument carries the movement control point that will cause lateral movement.

<table>
<thead>
<tr>
<th>RT</th>
</tr>
</thead>
</table>

- **RT**
  - This argument carries the route number that will cause lateral movement.

**Temporary Attribute**

<table>
<thead>
<tr>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYPASS</td>
</tr>
</tbody>
</table>

- **BYPASS**
  - This variable is an argument for event DIVERT. It carries the route number that will cause lateral movement.

<table>
<thead>
<tr>
<th>DIRC.ON.BT</th>
</tr>
</thead>
</table>

- **DIRC.ON.BT**
  - Indicates whether or not a vehicle is moving forward or backward on his route.
    - 0 Vehicle is moving in order of increasing MCP numbers along the route. (forward)
    - 1 Vehicle is moving in order of decreasing MCP numbers along the route. (backward)
FORMACODE

The formation number of the formation to be used by the platoon.

0 Not in formation, vehicle moves along route without offset.

HOLD.FORM

This unit attribute is a storage place, where the units formation number for the platoon (FORMACODE) can be placed when lateral route movement is appropriate.

HOME

This variable is an argument for event DIVERT. It carries the MCP number that will be cause lateral movement.

NEXT.MCP

Movement control point number (on the designated route) toward which the element is now moving.

0 End of route has been reached

ROUTE

Indicates the number of the route along which the element is travelling.

0 Not using a route

SIDE.STEPPER

This variable is an argument for event DIVERT. It carries the number of the entity to be moved laterally.

Brief Explanation
Lines 3-4: Directs the entity to go to a Movement Control Point on a new route.

Lines 5-6: Enables the entity to move straight to the MCP on this new route.

Line 7: Makes sure the entity is not moving backwards (away from the attack).

CODE

```
1 EVENT DIVERT GIVEN B, RT, MCP
2 DEFINE B, RT, MCP AS INTEGER VARIABLES
3 LET ROUTE(B) = RT
4 LET NEXT_MCP(B) = MCP
5 LET HOLD_FORM(B) = FORMACODE(B)
6 LET FORMACODE(B) = 0
7 LET DIRC.ON.HT(B) = 0
8 RETURN
9 END
```

263
14. **Routine MOMENTUM**

**Purpose**

Routine MOMENTUM allows a platoon member to move over the remaining cleared distance on a route through a minefield. This routine is called from routine HF.INTERNAL.

**Given Arguments**

- **Integer**
  - **B** - The pointer to the entity that needs a distance to travel in the minefield to keep up with his platoon.

- **Recursive Variable**
  - **Integer**
    - **TYK** - Pointer to an element in the platoon of the activating entity.

- **Recursive Variable**
  - **Real**
    - **MOV.LENG** - This is the distance left to be cleared on a route through the minefield.

- **Yielding Argument**
  - **Real**
    - **MOV.DIST** - This is the distance to the lead tank in the platoon from the activating entity.
Set

PLT.UIN (1-D)

This set is owned by the permanent entity platoon leader. It contains the list of temporary entities owned by this platoon.

Temporary Attribute  Integer

PLT

The number of the platoon to which the entity belongs.

SYS.TYPE

This represents the general class of the system of the entity.

1  tanks
2  mounted infantry
3  dismounted infantry
4  artillery
5  air
6  air defense
7  bunker
8  comm/ew/acq/intel
9  other

TBUMP.NUM

This unit attribute allows the entity to move in a very detailed manner as a member of his platoon when in an obstacle field.

Temporary Attribute  Real

FLD.BDY.DIST

This is the distance from the entity to the nearest field boundary.

FLD.INT.DIST

This attribute of an entity contains the distance to a pending field internal action.
Brief Explanation

Lines 4-8 Searches the platoon members to find the lead vehicle and his distance to clear the entire lane in the minefield.

Line 9 Obtains the distance that has already been cleared in front of the activating entity.

CODE

```
1 ROUTINE MOMENTUM GIVEN B, BUMP YIELDING MOV.DIST
2 DEFINE B, BUMP, TNK AS INTEGER VARIABLES
3 DEFINE MOV.DIST, MOV.LENG AS A REAL VARIABLE
4 FOR EVERY TNK IN PLT.Unit(PLT(B)) WITH SYS.TYPE(TNK) BE 3, DO
5      IF TBUMP.BUM(TNK) EQ BUMP
6          LET MOV.LENG = FLD.BDY.DIST(TNK) - FLD.INT.DIST(TNK)
7      ALWAYS
8     LOOP
9     LET MOV.DIST = FLD.BDY.DIST(B) - MOV.LENG
10    RETURN
11 END
```
15. **Routine PLW_ALIVE**

**Purpose**

Routine PLW_ALIVE checks the status of the platoon's plows. It also checks to see if a platoon member is a push tank or the lead vehicle participating in an offset. This routine is called from routines MF_DECISION, MF_ENTRY, and MF_INTERNAL.

**Given Argument**

Integer

B - The pointer to the entity that needs a status on the plows within his platoon.

**Yielding Argument**

Integer

STAT.PLOW

This variable gives the platoon's status of mine plows:

- 0-no plow available
- 1-plow available but not in use
- 2-plow being used
- 3-platoon is offsetting
- 4-platoon is pushing a dead plow tank

**Recursive Variable**

Integer

J - Receives the status of the platoon plows.

TNK

Pointer to an element in the platoon of the activating entity.

Set
PLT.UNIT (1-D)

This set is owned by the permanent entity platoon leader. It contains the list of temporary entities owned by this platoon.

Temporary Attribute Integer

KKILL

Indicates whether an entity has sustained a catastrophic kill.

0 No
1 Yes

HKILL

Indicates whether an entity has sustained a mobility kill.

0 No
1 Yes

PLOW.COND

This unit attribute gives the tank the ability to have a mine plow characteristic.

0-no plow available
1-plow available but not in use
2-plow being used
3-led vehicle in offset
4-tank that is pushing dead plow tank

PLT

The number of the platoon to which the entity belongs.

Brief Explanation

Lines 3-20 Searches the platoon for its highest plow condition/status. The platoon
can only have one such number at a time.

CODE

1 ROUTINE PLW_ALIVE GIVEN B YIELDING STAT.PLOW
2 DEFINE B.TNK,J,STAT.PLOW AS INTEGER VARIABLES
3 FOR EVERY TNK IN PLT_UNIT(PLT(B)) DO
4 IF KILL(TNK) EQ 1 OR SKILL(TNK) EQ 1
5 CYCLE
6 OTHERWISE
7 LET J = PLOW_COND(TNK)
8 IF J = 1
9 LET STAT.PLOW = 1
10 RETURN
11 OTHERWISE
12 IF J = 3
13 LET STAT.PLOW = 3
14 RETURN
15 OTHERWISE
16 IF J = 4
17 LET STAT.PLOW = 4
18 RETURN
19 OTHERWISE
20 LOOP
21 LET STAT.PLOW = 0
22 RETURN
23 END
16. **Event QUIK.MOVE**

**Purpose**

Event QUIK.MOVE updates the location of an entity. This event continues to update locations for a certain number of times at a prescribed interval. Event QUIK.MOVE is scheduled from routines MF.DECISION, MF.ENTRY, and GAP.DECISION. It is also scheduled from events QUIK.MOVE, WALL.BREACH, GAP.BREACH, HEAVY.JUNK, and GAP.JOCK.

**Given Arguments**

- **Integer**
  - **B** - This argument carries the number of the entity to be moved.
  - **COUNT** - This argument carries the number of iterations of scheduling left for this event to perform.

- **Real**
  - **INTERVAL** - A constant term that is set so that element locations are updated every five seconds during a small portion of the simulation.

**Routines Called**

- **LOC** - A routine used to determine whether movement is possible and to initiate a call to MOVE.

**Temporary Attribute**

- **Integer**

270
**KKILL**

Indicates whether an entity has sustained a catastrophic kill.

- 0 no
- 1 yes

**NKILL**

Indicates whether an entity has sustained a mobility kill.

- 0 no
- 1 yes

**NUM.POINTER**

This variable is an argument for event QUIK.MOVE. It carries the number of the entity to be moved.

**REP.NUMBER**

This variable is an argument for event QUIK.MOVE. It carries the number of iterations of scheduling left for this event to perform.

**Temporary Attribute** Real

**INT.TIME**

This variable is an argument for event QUIK.MOVE. It carries the time interval number when the next event is to be performed.

**Brief Explanation**

Lines 4-6 Updates the location of the entity, decrements the number of future schedulings, and schedules another event.
CODE

1 EVENT QUIK.MOVE GIVEN B, COUNT, INTERVAL
2 DEFINE B, COUNT AS INTEGER VARIABLES
3 DEFINE INTERVAL AS A REAL VARIABLE
4 CALL LOC(B)
5 IF COUNT GT 0 AND KILL(B) NE 1 AND gekill(B) NE 1
6 SCHEDULE A QUIK.MOVE GIVEN B, COUNT-1, INTERVAL IN
7 ALWAYS
8 RETURN
9 END
17. **Routine MINE.SCHED**

*Purpose*

Routine MINE.SCHED produces the distance that an activating minefield entity has to travel in order to encounter a mine. This routine is called from routines DROP.PLOW, MP.ENTRY, and MF.INTERNAL.

*Given Arguments*

<table>
<thead>
<tr>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLD- Pointer to the field in question.</td>
</tr>
<tr>
<td>VEH- Pointer to the element being moved.</td>
</tr>
</tbody>
</table>

*Global Variable*

<table>
<thead>
<tr>
<th>Real</th>
</tr>
</thead>
<tbody>
<tr>
<td>TARDIM (3-D)</td>
</tr>
</tbody>
</table>

Contains the target dimensions of all entity types in the simulation. This is indexed by SYS.TYPE and WPN.TYPE of the element.

*Permanent Attribute*

<table>
<thead>
<tr>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIT.BELT (1-D)</td>
</tr>
</tbody>
</table>

This PLATOON LEADER attribute stores the number of the next mine belt to be encountered in the minefield. It allows the platoon to move through a minefield as an organized unit.

*Recursive Variable*

<table>
<thead>
<tr>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNK</td>
</tr>
</tbody>
</table>

Pointer to an element in the platoon of the activating entity.
Recursive Variable Real

\textbf{ANGLE}

Set to the direction of movement of the entity.

\textbf{DRAW}

Contains the value of a random number from a \textbf{Uniform} \((0,1)\) distribution.

\textbf{EXPECT.DIST}

The reciprocal of the width times the density of the scatterable minefield.

\textbf{MF.DENSITY}

Is the number of mines in the minefield divided by the area of the field.

\textbf{THETA}

Is the direction of movement of the entity minus the orientation angle of the ellipse.

\textbf{WIDTH}

Takes on several values during this routine depending upon the situation.

- Trip wire length (TWL) - If infantry in an anti-personnel frag field.
- Two times the TWL plus the VW - If a vehicle is in an anti-personnel frag field.
- Vehicle width - If a vehicle is in an anti-tank field.

\textbf{Set}

\textbf{PLT.UNIT} \((1-D)\)

This set is owned by the permanent entity PLATOON LEADER. \textit{It contains the list of temporary entities owned by this platoon.}
Temporary Attribute  Integer

COLOR
Indicates the color of the element.

0  red (attacker)
1  blue (defender)

FLD.NO
Name of the field involved in any pending internal action.

NAME
The element number of the entity.

PLT
The number of the platoon to which the entity belongs.

SYS.TYPE
This represents the general class of the system of the entity.

1  tanks
2  mounted infantry
3  dismounted infantry
4  artillery
5  air
6  air defense
7  bunker
8  comm/ew/acq/intel
9  other

275
WHAT.BELT

This entity attribute keeps the updated belt number that is to be encountered next.

WPN.TYPE

Describes the specific system within the system code. For example, system type 1 is a tank and weapon type four for this system indicates an AVLB.

Temporary Attribute Real

ANGLE.FLD

Orientation angle in radians measured counterclockwise from the east to the major axis of the elliptical field.

AREA.FLD

This field attribute carries the area of the field in meters squared.

DIR.OF.MVMT

Indicates the entity's direction of movement measured in radians from east.

FLD.INT.DIST

This attribute of an entity contains the distance to a pending field internal action.

P1.FLD

The activation level of the minefield

<table>
<thead>
<tr>
<th>0</th>
<th>not activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>activated patterned minefield</td>
</tr>
<tr>
<td></td>
<td># of mines activated scatterable minefield</td>
</tr>
</tbody>
</table>

P2.FLD (Appendix D)
P3.FLD

Mine Type (1-11)

1  M70 Scatterable Anti-tank mine
2  M56 Scatterable Anti-tank mine
3  M21 Hand-emplaced Anti-tank mine
4  M15 Hand-emplaced Anti-tank mine
5  M19 Hand-emplaced Anti-tank mine
6  M14 Anti-personnel blast mine
7  M25 Anti-personnel blast mine
8  M16 Anti-personnel frag mine
9  ADAM Anti-personnel frag mine
10 M74 Anti-personnel frag mine
11 Claymore Anti-personnel frag mine

SAMAJ.FLD

The semi-major axis length of the elliptical field.

X.CURRENT

The X-coordinate for the entity as of the last movement update.

Y.CURRENT

The Y-coordinate for the entity as of the last movement update.

Brief Explanation

Lines 4-7  Checks if the minefield is still active. This is done primarily for the Anti-personnel minefields.

Lines 8-35  Simulates the Anti-personnel minefields.

Lines 9-19  The Anti-personnel blast minefield is modelled.
Line 10  Determines the minefield density in mines per square meter.

Lines 11-14  Calculates the distance to a mine encounter for a vehicle moving through an Anti-personnel blast minefield.

Lines 15-18  Calculates the distance to a mine encounter for a dismounted soldier moving through an Anti-personnel blast minefield.

Lines 20-35  Simulates the Anti-personnel frag minefield.

Lines 21-22  Sets the distance to a Claymore mine detonation.

Lines 24-26  Calculates the width of the lane made by a vehicle moving through an Anti-personnel frag minefield.

Line 28  Obtains the effective length of the AP frag mine in question.

Lines 30-32  Calculates the minefield density. This density and the effective lane width are used to obtain an average
mine encounter distance. The exponential distribution is then sampled to get the actual distance to the next mine.

Line 35 Decrements the number of mines in the minefield.

Lines 36-58 Models the effects of Anti-tank minefields.

Lines 38-42 Obtains a minefield density and the width of the activating entity. An average distance to mine encounter is calculated. A distance is sampled from an exponential distribution using the above average distance value.

Lines 43-58 Simulates a patterned/belted minefield.

Lines 44-47 Updates the next mine belt to be encountered by the platoon elements.

Lines 48-54 Calculates the distance to the next mine encounter using the distance between the belts and the orientation
of the minefield. Red/Attacking elements only are affected.

Lines 55-57 Simulates that blue elements know where the planned mine lanes are in their own minefields. These minefields have no effect on the defender.

Lines 59-61 Dismounted infantry will not detonate Anti-tank mines.

CODE

```plaintext
1 ROUTINE MINE.SCHED GIVEN VEH, FLD
2 DEFINE VEH, FLD, Tnk AS INTEGER VARIABLES
3 DEFINE DRAW, MP, DENSITY, EXPECT.DIST, ANGLE, THETA, WIDTH AS REAL VARIABLES
4 IF P1.FLD(FLD) = 0.0 "NOT ACTIVE ANY MORE"
5 LET FLD.INT.DIST(VEH) = RINF.C
6 RETURN
7 OTHERWISE
8 IF P3.FLD(FLD) GE 6 "ANTI-PERSONNEL MINES"
9 IF P3.FLD(FLD) LT 8 "MINE.TYPE=AP BLAST"
10 LET MP.DENSITY = P1.FLD(FLD)/AREA.FLD(FLD)
11 IF SYS.TYPE(VEH) NE 3
12 LET WIDTH = (TARDIN(SYS.TYPE(VEH), WPW.TYPE(VEH), 6)
13 LET EXPECT.DIST = 1.0/(WIDTH*MP.DENSITY)/3.0
14 LET FLD.INT.DIST(VEH) = EXPONENTIAL.F(EXPECT.DIST, 7)
15 ELSE
16 LET DRAW=UNIFORM.F(0., 1., 7)
17 LET DRAW=DE-DRAW
18 LET FLD.INT.DIST(VEH) = ((-2.5)*LOG.E.F(DRAW))/(MP.DENSITY**.12)
19 ALWAYS "MINE.TYPE = AP FRAG"
20 IF P3.FLD(FLD) EQ 11.0
21 LET FLD.INT.DIST(VEH) = .2*SAHAF.J.FLD(FLD)
22 ELSE
23 IF SYS.TYPE(VEH) NE 3
24 LET WIDTH = (TARDIN(SYS.TYPE(VEH), WPW.TYPE(VEH), 6)+
25 (P2.FLD(FLD)/.707)
26 ELSE
27 LET WIDTH = P2.FLD(FLD)/.707
28 ALWAYS
```

280
LET MF.DENSITY = P1.FLD(FLD)/AREA.FLD(FLD)
LET EXPECT.DIST = 1.0/(WIDTH*MF.DENSITY)
LET FLD.INT.DIST(VEH) = EXPONENTIAL.F(EXPECT.DIST,7)

ALWAYS
ALWAYS
SUBTRACT 1 FROM P1.FLD(FLD)
ELSE "'ANTI-TANK MINES
IF SYS.TYPE(VEH) NE 3 "'SOME TYPE OF VEHICLE
IF P3.FLD(FLD) LT 3
" P3.FLD = MINE.TYPE // SCATTERABLE MINES
LET MF.DENSITY = P1.FLD(FLD)/AREA.FLD(FLD)
"P1.FLD = NUMBER.MINES
LET WIDTH =
TARDIM(SYS.TYPE(VEH), WPN.TYPE(VEH), 6)
LET EXPECT.DIST = 1.0/(WIDTH*MF.DENSITY)
LET FLD.INT.DIST(VEH) = EXPONENTIAL.F(EXPECT.DIST,7)
ELSE "'A BELTED MINEFIELD
ADD 1 TO NEXT.BELT(PLT(VEH))
FOR EVERY TNK IN PLT.UNIT(PLT(VEH))
. WITH SYS.TYPE(TNK) NE 3
LET WHAT.BELT(TNK) = NEXT.BELT(PLT(VEH))
IF COLOR(VEH) = 0 "'RED FORCE WILL HIT MINES
IF DIR.OF.MVMT(VEH) LT 0.0
LET ANGLE = 360/RADIAN.C + DIR.OF.MVMT(VEH)
ELSE LET ANGLE = DIR.OF.MVMT(VEH)
ALWAYS LET THETA = ANGLE - ANGLE.FLD(FLD)
"P2.FLD = BELT.SEPARATION
LET FLD.INT.DIST(VEH) = P2.FLD(FLD)/ABS.F(SIN.F(THETA))
ELSE "'BLUE FORCE KNOWS WHERE THE LANES ARE
LET FLD.INT.DIST(VEH) = RINF.C
ALWAYS
ALWAYS
"'DISMOUNTED INFANTRY IN A AT MINEFIELD
LET FLD.INT.DIST(VEH) = RINF.C
ALWAYS
PRINT 1 LINE WITH NAME(VEH),X.CURRENT(VEH),
Y.CURRENT(VEH), THE, V
MAH.FLD(FLD), FLD.INT.DIST(VEH) AS FOLLOWS
MINE DET SCHED VEH **** LOC X **** **** T-**** FLD=****
RETURN
END

281
18. **Routine GAP_DECISION**

**Purpose**

Routine GAP_DECISION allows the entity and his platoon the options of going to the gap obstacle to attempt a bull through or breach, and the option to bypass or lane select, whichever is applicable. This routine is called from routine FLD.ACT.

**Given Arguments**

<table>
<thead>
<tr>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
</tbody>
</table>

Pointer to the entity that has entered the decision ellipse.

| FLD- pointer to the field in question. |

**Events Scheduled**

- DIVERT *(this appendix)*
- QUIK_MOVE *(this appendix)*
- TURN_AROUND *(this appendix)*

**Global Variable**

| FLD_POINTER (1-D) INTEGER |

This 1-dimensional array contains the temporary field pointers. This enables the user to access the obstacles by using the sequential order numbers given the fields on input.

| TRAF_CONTROL (3-D) REAL (Appendix A) |

282
Recursive Variable Integer

BRIDGE.STAT

Contains the status of the AVLB in the platoon.

COUNTER

Contains the number of five second move increments to get from the decision ellipse through the obstacle under the worst of conditions.

EQUIP.STAT

Contains the status of the blade tanks in the platoon.

I - Index for a do loop.

KNOW.LEVEL

Contains the value of the field attribute P6.FLD. This is the state of intelligence known about the obstacle.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No knowledge exists</td>
</tr>
<tr>
<td>2</td>
<td>Knowledge exists</td>
</tr>
<tr>
<td>3</td>
<td>Knowledge exists and a lane exists</td>
</tr>
</tbody>
</table>

OBS.FLD

The temporary field pointer of the obstacle.

OBS.NUM

The number of the obstacle to our front. This number is taken from the field's sequence on input.

Recursive Variable Real
DIST.AWAY

This is the distance to the obstacle from this decision ellipse.

INTERV

A constant term that is set so that element locations are updated every five seconds during a small portion of the simulation.

SHIN.OBS

Receives the value in meters of the semi-minor axis of the gap obstacle ellipse to the front.

Routines Called

JUNK.ALIVE (this appendix)

Temporary Attribute Integer

BYPASS

This variable is an argument for event DIVERT. It carries the route number that will cause lateral movement.

DIRC.ON.RT

Indicates whether or not a vehicle is moving forward or backward on his route.

- 0 Vehicle is moving in order of increasing MCP numbers along the route. (forward)
- 1 Vehicle is moving in order of decreasing MCP numbers along the route. (backward)

END AREA

The number which identifies the ending movement area for this entity.
FLD.AKT

Code describing the kind of action for pending internal actions.

FLD.NO

Name of the field involved in any pending internal action.

FORMACODE

The formation number of the formation to be used by the platoon.

0  not in formation, vehicle moves along route without offset.

HOLD.FORM

This unit attribute is a storage place, where the units formation number for the platoon (FORMACODE) can be placed when lateral route movement is appropriate.

HOME

This variable is an argument for event DIVERT. It carries the MCP number that will cause lateral movement.

NEXT.MCP

Movement control point number (on the designated Route) toward which the element is now moving.

0  end of route has been reached

NUM.POINTER

This variable is an argument for event QUICK.MOVE. It carries the number of the entity to be moved.
REP.NUMBER

This variable is an argument for event QUICK.MOVE. It carries the number of iterations of scheduling left for this event to perform.

ROUTE

Indicates the number of the route along which the element is travelling.

0 not using a route

SIDE.STEPPER

This variable is an argument for event DIVERT. It carries the number of the entity to be moved laterally.

START.AREA

The number which identifies the starting movement area for this entity.

SYS.TYPE

This represents the general class of the system of the entity.

1 tanks  
2 mounted infantry  
3 dismounted infantry  
4 artillery  
5 air  
6 air defense  
7 bunker  
8 coms/ew/acq/intel  
9 other

TWIST

This variable is an argument for event TURN.AROUND. It carries the number of the entity to be turned in the opposite direction.
**TYP.FLD**

Field type code

1 - a mandatory dismount field
2 - a minefield
3 - a minefield decision field
4 - a tank ditch
5 - a road crater
6 - a blown bridge (short-wet gap)
7 - a gap decision field

**Temporary Attribute Real**

**FLD.INT.DIST**

This attribute of an entity contains the distance to a pending field internal action.

**INT.TIME**

This variable is an argument for event QUIK.MOVE. It carries the time interval number when the next event is to occur.

**P1.FLD** (appendix d)

**P2.FLD** (Appendix D)

**P3.FLD** (Appendix D)

**P4.FLD** (Appendix D)

**P5.FLD** (Appendix D)

**P6.FLD** (Appendix D)

**SAHIN.FLD**

The semi-minor axis length of the elliptical field.
THE ENGINEER EFFECTS MODULE FOR THE STAR COMBAT MODEL. (U)
SPD

The entity's speed at the end of the most recent movement update.

XC.Fld

The X-coordinate of the center of an elliptical field.

YC.Fld

The Y-coordinate of the center of an elliptical field.

Brief Explanation

Lines 10-13 Interchanges the value of HOLD.FORM and FORMACODE. This allows the entity to pick up the route formation if applicable.

Lines 15-19 Checks if the decision ellipse is on a bypass around the obstacle and takes appropriate actions if it is.

Lines 20-23 Dismounted infantry are not affected by gap type obstacles.

Lines 24-25 Links to the knowledge attribute, P6.FLD, of the obstacle to the front.

Lines 26-27 Turns the entity around if he is coming back from the obstacle.
Lines 28-31  Allows the user to stipulate if a bypass is a viable option or if the entity is going to "bull through".

Lines 34-40  Sets the counter variable to the appropriate number of QUIK.MOVE iterations needed to get to the gap obstacle.

Line 41  Checks on the status of the platoon's AVLBS and dozer blade tanks.

Lines 42-57  Allows the entity that has knowledge of a lane in the obstacle to move to the lane, if his platoon is void of breaching equipment.

Lines 58-68  If the activating entity's platoon does not possess the required breaching equipment for the obstacle encountered and there is knowledge of the obstacle, then the entity can go to his bypass route and movement control point.

Line 70  Schedules five second movement updates for the entities going to the gap obstacle.
CODE

1 ROUTINE GAP-DECISION GIVEN A, FLD
2 DEFINE A, FLD.OBS.NUM.OBS.FLD.KNOW.LEVEL.EQUIP.STAT.
3 DEFINE INTERV, SHEIN.OBS., DIST.AWAY AS INTEGER VARIABLES
4 LET FLD.NO(A) = FLD
5 LET FLD.ACT(A) = FLD.FLD(FLD)
6 LET FLD.DST(A) = ROUTE(A)
7 LET FLD.NEXT(A) = NEXT.HCP(A)
8 LET INTERV = 5.0
9 IF HOLD.FORM(A) NE 999
10 LET FORNACO CODE(A) = HOLD.FORM(A)
11 LET HOLD.FORM(A) = 999
12 ALWAYS
13 ***LINK TO OBSTACLE KNOWLEDGE ATTRIBUTE***
14 LET OBS.NUM = INT.F(F3.FLD(FLD))
15 IF OBS.NUM = 0
16 "THIS ELLIPSE CIRCUMVENTS THE OBSTACLE"
17 LET FLD.INT.DIST(A) = RINF.C
18 RETURN
19 OTHERWISE
20 IF SYS.TYPE(A) = 3 "DISMOUNTED INFANTRY"
21 LET FLD.INT.DIST(A) = RINF.C
22 RETURN
23 OTHERWISE
24 LET OBS.FLD = FLD.POINTER(OBS.NUM)
25 LET KHOW.LEVEL = INT.F(P3.FLD(OBS.FLD))
26 IF DISC.ON.FT(A) = 1
27 SCHEDULE A TURN.AROUND GIVEN A NOW
28 IF P3.FLD(FLD) NE 0.0
29 "BYPASS EXISTS /FALSE-NO BYPASS (BULL-LANE)"
30 LET TRAF.CONTROL(OBS.NUM,ROUTE(A),1) = NEXT.HCP(A)
31 ALWAYS
32 IF P3.FLD(FLD) LT 0.0
33 LET FLD.INT.DIST(A) = RINF.C
34 ELSE
35 LET FLD.INT.DIST(A) = DIST.AWAY
36 LET SHEIN.OBS = SABIN.FLD(OBS.FLD)
37 LET DIST.AWAY = SQRT.F((IC.FLD(FLD) - IC.FLD(OBS.FLD))^2 + (IC.FLD(FLD) - IC.FLD(OBS.FLD))^2)
38 LET COUNTER = INT.F((INTERV + DIST.AWAY + SHEIN.OBS)/SPD(A))/INTERV
39 LET P3.FLD(FLD) = COUNTER
40 ALWAYS
41 CALL JUNK.ALIVE GIVEN A YIELDING EQUIP.STAT,BRIDGE.STAT
42 IF KNOW.LEVEL = 3
43 IF P3.FLD(OBS.FLD) EQ "TANK BICHR" AND
44 IF EQUIP.STAT EQ 1 AND BRIDGE.STAT EQ 1
45 SCHEDULE A DIVERT(A)
46 IF TRAP.CONTROL(OBS.NUM,1,3) EQ RINF.C
47 AND TRAP.CONTROL(OBS.NUM,1,2) NE 0.0
48 SCHEDULE A DIVERT(A)
49 RETURN
50 ELSE
51 OTHERWISE
52 "LANE EXISTS-NOT PRIVY TO ITS LOCATION"
53 RETURN
54 "LANE EXISTS-NOT PRIVY TO ITS LOCATION"
55 ALWAYS

290
56 ALWAYS
57 IF KNOW.LEVEL = 2 ""WE KNOW THE OBSTACLE EXISTS
58 IF BRIDGE.STAT ME 1 ""NO BRIDGING EQUIPMENT
59 IF TYP.FLDJOBS.FLD) EQ 6 OR EQUIP.STAT ME 1
60 IF P1.FLD(FLD) NE 0.0:""A BYPASS EXISTS
61 SCHEDULE A DIVERT (A, INT.F(P1.FLD.FLD)).
62 INT.F(P2.FLD.FLD)) NOW
63 RETURN
64 OTHERWISE
65 ALWAYS
66 ALWAYS
67 ALWAYS
68 LET COUNTER = INT.F(P6.FLD.FLD))
69 SCHEDULE A QUICK.MOVE GIVEN A,COUNTER,INTERV IN
70 INTERV UNITS
71 RETURN
72 END
19. **Routine JUNK-ALIVE**

Purpose

Routine JUNK-ALIVE checks the status of the platoon dozer blade tanks and vehicle launched bridges. This routine is called by routines GAP.DECISION, GAP.ENTRY, and events HEAVY.JUNK, and GAP.JOCK.

Given Arguments Integer

B

Pointer to the entity who's platoon status is being checked.

Yielding Arguments Integer

BRIDGE.DESC

Contains the status of the AVLBs in the platoon.

EQUIP.DESC

Contains the status of the blade tanks in the platoon.

Recursive Variable Integer

TNS

Pointer to an element in the platoon of the activating entity.

Set

PLT.UNIT (1-D)

This set is owned by the permanent entity PLATOON LEADER. It contains the list of temporary entities owned by this platoon.
Temporary Attribute Integer

BLADE.COND

This unit attribute gives the tank the ability to have a dozer blade characteristic.

0 - no blade available
1 - blade available but not in use
2 - blade being used or vehicle is self breaching

KILL

Indicates whether an entity has sustained a catastrophic kill.

0  no
1  yes

KILL

Indicates whether an entity has sustained a mobility kill.

0  No
1  yes

PLT

The number of the platoon to which the entity belongs.

SYS.TYPE

This represents the general class of the system of the entity.

1  tanks
2  mounted infantry
3  dismounted infantry
4  artillery
5  air
6  air defense
7  bunker
8  comm/ei/acq/intel
9  other

UPN.TYPE

Describes the specific system within the system.
code. For example, system type one is a tank and weapon type four for this system indicates an AVLB.

Brief Explanation

Lines 3-14 Checks the platoon for the presence of alive AVLBs and dozer blade tanks.

CODE

```
1 ROUTINE JUNK.ALIVE GIVEN B
2   XIELDING EQUIP.STAT,BRIDGE.STAT
3 DEFINE B, EQUIP.STAT, THK, BRIDGE.STAT
4   L1 INTEGER VARIABLES
5 FOR EVERY THK IN PLT.UNIT(PLT(B)), DO
6     IF KILL(TNK) NE 1 AND KILL(TNK) NE 1
7         IF BLADE.COMB(TNK) GE 1
8         LET EQUIP.STAT = 1
9     ALWAYS
10        IF SYS.TYPE(TNK) EO 1
11            IF WPN.TYPE(TNK) = 4 OR WPN.TYPE(TNK) = 5
12                LET BRIDGE.STAT = 1
13            ALWAYS
14            ALWAYS
15        ALWAYS
16 LOOP
17 RETURN
18 END
```
20. **Routine GAP-ENTRY**

**Purpose**

Routine GAP-ENTRY allows an entity to take the appropriate actions when encountering a gap type obstacle. This routine is called from Routine FLD.ACT.

**Given Arguments**

Integer

- **A** - Pointer to the entity that just entered the gap obstacle.

**FLD** - Pointer to the field in question.

**Events Scheduled**

- GAP.JOCK (this appendix)
- HALT (this appendix)
- HEAVY.JUNK (this appendix)
- TURN.AROUND (this appendix)

**Global Variable**

- **TRAP.CONTROL** (3-D) (Appendix A)

**Recursive Variable**

- **BRIDGE.STAT**

Contains the status of the AVLBS in the platoon.
EQUIP. STAT
Contains the status of the blade tanks in the platoon.

KNOWN.LEVEL
Contains the value of the field attribute P6.FLD. This is the state of intelligence known about the obstacle.

0  no knowledge exists
2  knowledge exists
3  knowledge exists and a lane exists

OBS. NUM
Receives the value of the field attribute HAH.FLD. This is the sequential id number in order of field creation.

TKR
Pointer to an element in the platoon of the activating entity.

Recursive Variable    Real

SIT.REP.TIME
This is the time between status checks on the breaching operation.

SIN
This variable is set to the semi-minor axis length of the elliptical field in meters.

Routines Called

HIDER
A routine used to determine the micro-terrain elevation for a selected element.
Set

PLT.UNIT (1-D)

This set is owned by the permanent entity PLATOON LEADER. It contains the list of temporary entities owned by this platoon.

Temporary Attribute Integer

BLADE.COND

This unit attribute gives the tank the ability to have a dozer blade characteristic.

DEPNUM

The current position or activity of an element.

1  full defilade
2  turret defilade
3  firing defilade
4  half vehicle defilade
5  moving (defilade determined by the terrain model)
6  reached final area in movement

FLD.AKT

Code describing the kind of action for pending internal actions.

FLD.FORM

This unit attribute gives the entity the ability to change formations when encountering an obstacle.

FLD.NO

Name of the field involved in any pending internal action.
1& 3. FLD
This is the sequential id number in order of field creation.

PLT
The number of the platoon to which the entity belongs.

ROUTE
Indicates the number of the route along which the element is travelling.
0 not using a route

SABOT.SHOOTER
This variable is an argument for event GAP.JOCK. It carries the number of the activating entity.

SLOW.DOWN
This variable is an argument for event HALT. It carries the number of the entity to be stopped.

STATUS
This variable is an argument for event HEAVY.JUNK. It carries the number of the entity to be updated.

SYS.TYPE
This represents the general class of the system of the entity.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>tanks</td>
</tr>
<tr>
<td>2</td>
<td>mounted infantry</td>
</tr>
<tr>
<td>3</td>
<td>dismounted infantry</td>
</tr>
<tr>
<td>4</td>
<td>artillery</td>
</tr>
<tr>
<td>5</td>
<td>air</td>
</tr>
<tr>
<td>6</td>
<td>air defense</td>
</tr>
<tr>
<td>7</td>
<td>bunker</td>
</tr>
<tr>
<td>8</td>
<td>coms/cw/acq/intel</td>
</tr>
<tr>
<td>9</td>
<td>other</td>
</tr>
</tbody>
</table>

298
TWIST

This variable is an argument for event TURN.AROUND. It carries the number of the entity to be turned in the opposite direction.

TYP.FLD

Field type code.

1 - a mandatory dismount field
2 - a minefield
3 - a minefield decision field
4 - a tank ditch
5 - a road crater
6 - a blown bridge (short-wet gap)
7 - a gap decision field

WPN.TYPE

Describes the specific system within the system code. For example, system type 1 is a tank and weapon type four for this system indicates an AVLB.

Temporary Attribute Real

FLD.INT.DIST

This attribute of an entity contains the distance to a pending field internal action.

P1.FLD

This gap field attribute is a flag that specifies activation state.

0 not activated yet
1 this is an active gap obstacle

P6.FLD

This is the state of intelligence known about the obstacle.

0 no knowledge exists
2 knowledge exists
3 knowledge exists and a lane exists
P7.FLD

This minefield attribute contains the lane formation number.

SA.MIN.FLD

This field attribute is the semi-minor axis length of the elliptical field in meters.

Brief Explanation

Lines 16-19 If the obstacle is not activated yet then the activating entity is allowed to move freely through this field.

Lines 20-23 Dismounted infantry soldiers are not affected by gap type obstacles.

Lines 24-27 When the entity is 200 meters from a large hole in the ground, he knows the gap exists.

Lines 28-36 If this route is on a cleared lane through the obstacle, change the formation of the platoon to a column and continue at present speed toward the crossing site.

Line 37 Checks the platoon AVLB and blade tank status.

Lines 41-43 If an entity has to "bull through" but a breaching operation is already
underway or his route, then the entity will stop and cover this operation by direct fire.

Lines 44-46
When an entity has to "bull through" and no breaching operations are ongoing for his route, then the entity will move to the gap and attempt a self breach.

Lines 47-56
If the entity has a bypass route but his platoon is void of appropriate breaching equipment for this obstacle, then the activating entity and his platoon will turn around and move back toward the decision ellipse.

Lines 60-65
If the entity is an AVLX, then it will proceed to the gap to attempt a breach/span. The flag for an ongoing breach is set to the entity’s platoon number.

Lines 66-70
If the entity is a blade tank and this obstacle is a road crater or tank ditch, then the tank will proceed to the gap to attempt a
breach. The flag for an ongoing breach is set to the blade tank's platoon number.

Lines 71-72 The vehicles in the platoon not equipped with breaching devices will stop and cover the gap clearing operation by direct fire from half defilade. A situation report on this operation is scheduled in the future.

CODE

1 ROUTINE GAP.ENTRY GIVEN A, FLD
2 DEFINE A, FLD, KNOW. LEVEL, OBS. NUM, TANK, EQUIP. STAT,
3 BRIDGE, STAT as INTEGER VARIABLES
4 DEFINE SIT. REP. TIME, SHIN as REAL VARIABLES
5 SUBSTITUTE THESE 3 LINES FOR SUPPORT.SHOOT
6 SCHEDULE A HALT GIVEN A NOW
7 LET KNOW.(A) = 0 "HALF DEPILADE" CALL RIDER(A)
8 LET FLD.INT.DIST(A) = 50.0 "A MUST DISTANCE
9 LET FLD.NG.(A) = FLD
10 LET FLD.ACT.(A) = TYP. FLD.(FLD)
11 LET KNOW. LEVEL = INT.B(PO. FLD.(FLD))
12 LET OBS. NUM = NUM. FLD.(FLD)
13 LET SIT. REP. TIME = 15.0 "THE TIME OF STATUS CHECKS
14 LET TRAFFIC. CONTROL(OBS. NUM, ROUTE(A), 1) = ROUTE(A)
15 LET SHIN = SHIN. FLD.(FLD)
16 IF F1. FLD.(FLD) EQ 0.0 "NOT ACTIVATED YET
17 LET FLD.INT.DIST(A) = RINF.C
18 RETURN
19 OTHERWISE
20 IF SYSTYPE(A) = 3 "DISMOUNTED INFANTRY
21 LET FLD.INT.DIST(A) = RINF.C
22 RETURN
23 OTHERWISE
24 IF KNOW. LEVEL LE 1 "NO KNOWLEDGE EXISTS
25 LET KNOW. LEVEL = 2
26 LET F1, FLD.(FLD) = 2
27 ALWAYS IF KNOW. LEVEL EQ 3
28 IF TYP. FLD.(FLD) NE 4 "NOT A TANK DITCH" OR
29 TYP. FLD.(FLD) EQ 4 "TANK DITCH" AND
30 TRAFFIC. CONTROL(OBS. NUM, ROUTE(A) 3) EQ RINF.C
31 LET FLD.FORM(A) = INT.B(09. FLD.(FLD))
32 RETURN
33 OTHERWISE
34 ALWAYS CALL JUNK. ALIVE GIVEN A YIELDING
IF BRIDGE, STAT NE 1 'NO BRIDGING EQUIPMENT'
  IF TYP, FLD(FLD) EQ 6 OR EQUIP, STAT NE 1
  LET TRAP, CONTROL(OBS, NUM, ROUTE(A, 2)) = 0.0
  IF TRAP, CONTROL(OBS, NUM, ROUTE(A, 4)) GT 0
    SUPPORT, SHOOT
    SCHEDULE A GAP, J OCK GIVEN A IN
    SIT, REP, TIME UNITS
  ELSE
    LET FLD, INT, DIST(A) = SHIN - 50.0
    ALWAYS
  ELSE 'A BYPASS EXISTS/ B, B MUST HAVE ONE
    SCHEDULE A TURN, AROUND GIVEN A NOW
    FOR EVERY TK IN PLT, UNIT(PLT(A))
    WITH SYS, TYPE(TNK) NE 3, DO
    IF FLD, EO(TNK) NE FLD
      SCHEDULE A TURN, AROUND (TNK) NOW
    ALWAYS
    LOOP
    LET FLD, INT, DIST(A) = RINF, C
    ALWAYS
    RETURN
    OTHERWISE
    ALWAYS
    IF SYS, TYPE(A) EQ 1 AND
      WPN, TYPE(A) = 8 OR WPN, TYPE(A) = 5
      LET FLD, INT, DIST(A) = SHIN - 50.0
      LET TRAP, CONTROL(OBS, NUM, ROUTE(A, 4)) = PLT(A)
    RETURN
    OTHERWISE
    IF TYP, FLD(FLD) NE 6 AND BLADE, COND(A) EQ 1
      LET FLD, INT, DIST(A) = SHIN - 50.0
      LET TRAP, CONTROL(OBS, NUM, ROUTE(A, 4)) = PLT(A)
    RETURN
    OTHERWISE
    SUPPORT, SHOOT
    SCHEDULE A HEAVY, JUNK GIVEN A IN SIT, REP, TIME UNITS
    RETURN
    END

303
21. **Routine GAP.INTERNAL**

**Purpose**

Routine GAP.INTERNAL gives the activating entity the ability to take the appropriate actions at the location of a gap type obstacle. This routine is called from routine FLD.ACT.

**Given Arguments**

- Integer
  - A - Pointer to the entity that just activated this gap internal action.

**Events Scheduled**

- DITCH.KILL (this appendix)
- GAP.BREACH (this appendix)
- HALT (this appendix)
- HEAVY.JUNK (this appendix)
- STAND.TO (this appendix)
- WALL.BREACH (this appendix)

**Global Variable**

- Real
  - TRAP.CONTROL (3-D) (Appendix A)
Recursive Variable Integer

BRIDGE.STAT
Contains the status of the AVLBS in the platoon.

EQUIP.STAT
Contains the status of the blade tanks in the platoon.

FLD- Pointer to the field in question.

KNOW.LEVEL
Contains the value of the field attribute P6.FLD. This is the state of intelligence known about the obstacle.

- 0 no knowledge exists
- 2 knowledge exists
- 3 knowledge exists and a lane exists

OBS.NUM
Receives the value of the field attribute NA.FlD. This is the sequential id number in order of field creation.

TUK
Pointer to an element in the platoon of the activating entity.

Recursive Variable Real

DRAW
Contains the value of a random number from a Uniform(0,1) distribution.
**SIT.REP.TIME**

This is the time between status checks on the breaching operation.

**Routines Called**

**HIDER**

A routine used to determine the micro-terrain elevation for a selected element.

**Temporary Attribute Integer**

**BAGGED.BOY**

This variable is an argument for event STAND.TO. It carries the number of the entity to be moved.

**BLADE_COND**

This unit attribute gives the tank the ability to have a dozer blade characteristic.

- 0-no blade available
- 1-blade available but not in use
- 2-blade being used or vehicle is self breaching

**BREACHER**

This variable is an argument for event GAP.BREACH. It carries the number of the entity that made the breach.

**DEFNUM**

The current position or activity of an element.

- 1 full defilade
- 2 first defilade
- 3 firing defilade
- 4 half vehicle defilade
- 5 moving (defilade determined by the terrain model)
- 6 reached final area in movement
**FLD.FORM**

This unit attribute gives the entity the ability to change formations when encountering an obstacle.

**FLD.NO**

Name of the field involved in any pending internal action.

**GULLY.CAT**

This variable is an argument for event DITCH.KILL. It carries the number of the entity that is stuck in the gap.

**HOLE**

This variable is an argument for event GAP.BREACH. It carries the number of the field in question.

**HAB.PLD**

This is the sequential id number in order of field creation.

**RAMMER**

This variable is an argument for event WALL.BREACH. It carries the number of the entity that made the self breach.

**ROUTE**

Indicates the number of the route along which the element is travelling.

0 not using a route

**SLOW.DOWN**

This variable is an argument for event HALT. It carries the number of the entity to be stopped.
STATUS

This variable is an argument for event HEAVY.JUNK. It carries the number of the entity to be updated.

SYS.TYPE

This represents the general class of the system of the entity.

- tanks
- mounted infantry
- dismounted infantry
- artillery
- air
- air defense
- bunker
- comm/ew/acq/intel
- other

TEAR.DOWN

This variable is an argument for event WALL.BREACH. It carries the number of the field in question.

TYP.FLD

Field type code.

1 - a mandatory dismount field
2 - a minefield
3 - a minefield decision field
4 - a tank ditch
5 - a road crater
6 - a blown bridge (short-wet gap)
7 - a gap decision field

WPN.TYPE

Describes the specific system within the system code. For example, system type 1 is a tank and weapon type four for this system indicates an AMLB.

Temporary Attribute: Real

308
FLD.INT.DIST

This attribute of an entity contains the distance to a pending field internal action.

FSPEED.FAC

This unit attribute gives the entity the ability to slow down because of obstacle encounter.

P6.FLD

Knowledge level of this obstacle.

0  no knowledge exists
2  knowledge exists
3  knowledge exists and a lane exists

P7.FLD

This minefield attribute contains the lane formation number.

P8.FLD

Lane speed factor.

P9.FLD

AVLB activation delay time in seconds.

P10.FLD

Tank dozer breach time in seconds.

P11.FLD

Tank self breach time in seconds.
Probability of a self breach getting stuck in ditch.

Brief Explanation

Lines 10-18  The entity's speed is reduced and he is put into a column formation. This simulates the funneling effect of the actual crossing.

Lines 19-42  Simulates the self-breach option against road craters and tank ditches.

Lines 23-26  When an entity has to self breach a tank ditch, he checks if a breach has been accomplished elsewhere. If a lane exists then the entity's speed is reduced substantially to simulate his movement to another existing route/lane. His formation is also changed to column.

Lines 28-40  Simulates the actual self breach. The entity is stopped, he is put in gun tube defilade (in ditch), and he is prevented from firing in this
configuration. An input parameter is Monte Carlo'ed against, to determine if the entity is going to be a mobility casualty. When not stuck in the ditch the entity has a breach scheduled to occur in the future.

Line 43

Schedules a breaching operation status update for the AVLB's and blade tank's attempting to breach this obstacle.

Lines 44–48

Stops the blade tank and starts his breaching effort. This tank is put in half defilade (blade and ditch shading), and he is prevented from firing. A successful breach is scheduled to occur in the future.

Lines 49–51

Stops the AVLB and starts his breaching effort. A successful breach/span is scheduled to occur in the future.

CODE

1 ROUTINE GAP, INTERNAL GIVEN A
2 DEFINE A, P10, POP. LEVEL, OBS, NUM, EQUIP-STAT,
3 BRIDGE-STAT, TNK AS INTEGER VARIABLES
4 DEFINE DRAW, SIT, REP. TIME AS REAL VARIABLES
LET PLO = LO.
LET KNOW.LEVEL = INT. F(P6. FLD(FLD))
LET OBS.NUM = NAM. FLD(FLD)
LET SIT.REP.TIME = 15.0 'TIME TO UPDATE STATUS
LET FLD.INT.DIST(A) = 50.0 'A MUST DISTANCE
IF KNOW.LEVEL EQ 3
IF TYP.FLD(FLD) NE 4 OR (TYP.FLD(FLD) EQ 4 AND
TSP. CONTROL(OBS. NUM, ROUTE(A), 3) EQ 2 INF.
LET FLD.INT. DIST(A) = RINF.C
LET FLD.FORM(A) = INT. F(P7. FLD(FLD))
RETURN
ELSE 'WE NEED TO BREACH A LANE THRU
IF TYP.FLD(FLD) NE 6
IF BLADE. COND(A) NE 1
IF SYS. TYPE(A) NE 1 OR (SYS. TYPE(A) EQ 1 AND
(WPN. TYPE(A) NE 4 AND WPN. TYPE(A) NE 5))
LET FSP. FLD(A) = P8. FLD(FLD) * 0.5
LET FLD.INT. DIST(A) = RINF.C
LET FLD.FORM(A) = INT. F(P7. FLD(FLD))
ELSE SCHEDULE A HALT GIVEN A NOW
LET DEF. NUM(A) = 3 'GUN' CALL HIDER(A)
LET BLADE. COND(A) = 2 'STOP SHOOTING
LET DRAW = UNIFORM. F(0. - 1., 7)
IF P12. FLD(FLD) GT DRAS
SCHEDULE A DITCH. KILL(A) IN
(P11. FLD(FLD)/2.0) UNITS
ELSE SCHEDULE A WALL. BREACH(A, FLD) IN
P11. FLD(FLD) UNITS
SCHEDULE A STAND. TO(A) IN
(P11. FLD(FLD) + 1) UNITS
ALWAYS
RETURN
END
22. **Routine GAP.LEAVE**

**Purpose**

Routine GAP.LEAVE performs the gap obstacle exit actions for the activating/moving entity. This routine is called from routine FLD.ACT.

**Given Arguments**

- **Integer**
  - A - Pointer to the entity that just activated this gap exit action.

**Global Variable**

- **Real**
  - TRAF.CONTROL (3-D) (Appendix A)

**Recursive Variable**

- **Integer**
  - FLD- Pointer to the field in question.

**Temporary Attribute**

- **Integer**
  - OBS.NUM

  Receives the value of the field attribute #AM.FLD. This is the sequential id number in order of field creation.

- **Integer**
  - BLADE.COMD

  This unit attribute gives the tank the ability to have a dozer blade characteristic.

  - 0 - no blade available
  - 1 - blade available but not in use
  - 2 - blade being used or vehicle is self-breaching

- **Integer**
  - DIREC.ON. RT

  Indicates whether or not a vehicle is moving forward or backward on his route.
0 Vehicle is moving in order of increasing HCP numbers along the route. (forward)
1 Vehicle is moving in order of decreasing HCP numbers along the route. (backward)

FLD.FORM

This unit attribute gives the entity the ability to change formations when encountering an obstacle.

FLD.NO

Name of the field involved in any pending internal action.

NAM.FLD

This is the sequential id number in order of field creation.

ROUTE

Indicates the number of the route along which the element is travelling.

0 not using a route

SYS.TYPE

This represents the general class of the system of the entity.

1 tanks
2 mounted infantry
3 dismounted infantry
4 artillery
5 air
6 air defense
7 bunker
8 comm/sw/acq/intel
9 other

Temporary Attribute   Real

314
This unit attribute gives the entity the ability to slow down because of obstacle encounter.

This gap field attribute is a flag that specifies activation state.

- 0: Not activated yet
- 1: This is an active gap obstacle

Knowledge level of this obstacle.

- 0: No knowledge exists
- 2: Knowledge exists
- 3: Knowledge exists and a lane exists

Brief Explanation

- Lines 3-5: Dismounted infantry are not affected.
- Lines 7-9: Inactivated obstacles have no effects.
- Lines 10-12: Prevents unplanned entities from affecting the situation.
- Lines 16-18: Changes the knowledge level of this obstacle. This transmits to follow-on units that a breach/span exists in this obstacle.
CODE

1 ROUTINE GAP.LEAVE GIVEN A
2 DEFINE A.FLD.OBS.NUM AS INTEGER VARIABLES
3 IF SYS.TYPE(A) = 3 'DISMOUNTED INFANTRY
4 RETURN
5 OTHERWISE
6 LET FLD = FLD.NO(A)
7 IF P1.FLD(FLD) EQ 0.0 'NOT ACTIVATED YET
8 RETURN
9 OTHERWISE
10 IF BLADE_COND(A) GE 2
11 RETURN
12 OTHERWISE
13 LET OBS.NUM = MAN.FLD(FLD)
14 LET FSP.FAC(A) = 1.0
15 IF DIRECTION.BT(A) = 0
16 LET P6.FLD(FLD) = 3
17 ALWAYS
18 RETURN
19 END

316
23. **Event DITCH.KILL**

**Purpose**

Event DITCH.KILL inflicts a mobility kill on the entity that got stuck in the tank ditch or road crater while attempting a self breach. This event is scheduled from routine GAP.INTERNAL.

**Given Arguments**

* Integer

B - The pointer to the entity that is stuck in the gap obstacle.

**Global Variable**

* Integer

DAM.ARRAY (1-D)

Contains the possible values for HIT.STATE.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NDAM</td>
</tr>
<tr>
<td>2</td>
<td>NDAM</td>
</tr>
<tr>
<td>3</td>
<td>NDAM</td>
</tr>
<tr>
<td>4</td>
<td>NDAM</td>
</tr>
<tr>
<td>5</td>
<td>DEATH</td>
</tr>
<tr>
<td>6</td>
<td>MISS</td>
</tr>
</tbody>
</table>

**DAMAGE.NUM**

Indicates the damage status of an entity after having a round impact on or near it.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hit but already HF killed</td>
</tr>
<tr>
<td>2</td>
<td>Mobility damage</td>
</tr>
<tr>
<td>3</td>
<td>Firepower damage</td>
</tr>
<tr>
<td>5</td>
<td>Mobility and firepower damage</td>
</tr>
<tr>
<td>6</td>
<td>Catastrophic Kill</td>
</tr>
<tr>
<td>7</td>
<td>Miss</td>
</tr>
</tbody>
</table>

**DEAD.ATKR**

Tallies the number of red casualties.
ONDISK

Indicates whether or not the user desires the shot list and final attribute list to user specified disk files.

0 no, paper output only
1 yes

YES

Indicates a value of 1. Used in conditional statements.

Recursive Variable Integer

WHOCALLED

A flag used by several routines to determine appropriate actions and executions.

Recursive Variable Real

EFKILL

The probability of at least a firepower kill.

EMKILL

The probability of at least a mobility kill.

EMEFKILL

The union of EMKILL and EFKILL.

KAYKILL

The probability of a catastrophic kill.

Routines Called
ATRIT

A Routine which assesses the attrition against a target which has been hit. It checks for a catastrophic kill, increments mobility and firepower damage, and checks the FKILL, MKILL, and BKILL levels.

HIDER

A routine used to determine the micro-terrain elevation for a selected element.

Temporary Attribute Alpha

HIT.STATE

An alpha variable indicating whether or not an element is alive or dead.

Temporary Attribute Integer

COLOR

Indicates the color of the element.

<table>
<thead>
<tr>
<th>Integer</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>red (attacker)</td>
</tr>
<tr>
<td>1</td>
<td>blue (defender)</td>
</tr>
</tbody>
</table>

DEFNUM

The current position or activity of an element.

<table>
<thead>
<tr>
<th>Integer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>full defilade</td>
</tr>
<tr>
<td>2</td>
<td>target defilade</td>
</tr>
<tr>
<td>3</td>
<td>firing defilade</td>
</tr>
<tr>
<td>4</td>
<td>half vehicle defilade</td>
</tr>
<tr>
<td>5</td>
<td>moving (defilade determined by the terrain model)</td>
</tr>
<tr>
<td>6</td>
<td>reached final area in movement</td>
</tr>
</tbody>
</table>

FIRED.AT

Indicates the total number of rounds fired at an entity.
FKILL
Indicates whether an entity has sustained a firepower kill.
  0  no
  1  yes

GULLY.CAT
This variable is an argument for event DITCH.KILL. It carries the number of the entity that is stuck in the gap.

K.HIT
Indicates the number of hits sustained by an entity which were sufficient to cause a catastrophic kill.

KKILL
Indicates whether an entity has sustained a catastrophic kill.
  0  no
  1  yes

MKILL
Indicates whether an entity has sustained a simultaneous mobility and firepower kill.
  0  no
  1  yes

BKILL
Indicates whether an entity has sustained a mobility kill.
  0  no
  1  yes

NAME
The element number of the entity.
NUM.HIT
Indicates the total number of hits sustained by an entity. This includes no damage hits.

ROUTE
Indicates the number of the route along which the element is travelling.
0 not using a route

WPN.TYPE
Describes the specific system within the system code. For example, system type 1 is a tank and weapon type four for this system indicates an AVLB.

Temporary Attribute Real

F.D
Indicates the accumulated percentage of firepower damage sustained by the entity.

M.D
Indicates the accumulated percentage of mobility damage sustained by the entity.

SPD
The entity's speed at the end of the most recent movement update.

X.CURRENT
The X-coordinate for the entity as of the last movement update.

Y.CURRENT
The Y-coordinate for the entity as of the last movement update.
Z. CURRENT

The elevation for the entity as of the last movement update.

Brief Explanation

Lines 5-7  The entity is already dead, do nothing.

Lines 8-33 The entity is a mobility casualty. This happened half-way through his self breach/bull through effort in the ditch. The killer/victims scoreboard is updated and the entity is put in turret defilade (bottom of ditch); still unable to fire.

CODE

EVENT DITCH.KILL GIVEN B
DEFINE B,H5OCALLED AS INTEGER VARIABLES
DEFINE ERP1LL, EKILL, EMTFILL,KAYFIIL AS REAL VARIABLES
LET H5OCALLED = 1
IF KILL(B) EQ 1
RETURN
OTHERWISE
LET ERP1LL = 1.0
LET EKILL = 0.0
LET EMTFILL = 1.0
LET KAYFIIL = 0.0
CALL AT5IT(B, B, ERP1LL, EKILL, EMTFILL, KAYFIIL, H5OCALLED)
LET HIT.STATE(B) = DAB.ARRAY(DAMAGE.NUM)
USE UNIT 6 FOR OUTPUT
START NEW LINE
'UP'
WRITE NAME(B), TYPE(B), STATE(B), CURRENT(B),
E.P(5), E.KILL(B), E.MT(KILL(B), E.KILL(B),
KILL(B), N.KILL(B), N.HIT(B)
AS 6, 6, 1, 1, 2, 1, 4, 1, 3, 5, 5, 5, 22, 3, 1, 7,
S 3, 3, 3, 6, 3, 1, A 4, 2 D(5,2), 2, 1, 2, 5, I 3
23 IF ONSK=YES AND WRITE.V=6 USE UNIT 10 FOR OUTPUT
24 START NEW LINE GO TO UP ELSE
25 IF COLOR(B) EQ 0 AND ([DAMAGE.NUM EQ 5 OR ([DAMAGE.NUM EQ 4 OR 26 DAMAGE.NUM EQ 3 OR DAMAGE.NUM EQ 2]) AND 27 BADKILL(B) EQ 1])
28 AND ( [DAMAGE.NUM NE 5 OR BADKILL(B) NE 1) AND
29 HIT.STATE(B) NE "DEAD" AND
30 HIT.STATE(B) NE "ADED"
31 ADD 1 TO DEAD.ATK ALWAYS
32 IF HIT.STATE(B) NE "DEAD" AND HIT.STATE(B) NE "ADED"
33 LET HIT.STATE(B) = "" ALWAYS
34 START NEW LINE
35 LET DEFNUM(B) = 2 ' 'TURRET DEFILADE' ' CALL HIDER(B)
36 RETURN
37 END
24. **Event WALL.BREACH**

**Purpose**

Event WALL.BREACH simulates the successful self breach of a road crater or tank ditch. This event is scheduled from routine GAP.INTERNAL.

**Given Arguments**

**Integer**

- **B** - The pointer to the entity that has self breached the gap obstacle.

- **FLD** - Pointer to the field in question.

**Events Scheduled**

- **QUIK.MOVE** (this appendix)

**Global Variable**

**Real**

- **LIN.SPDS** (3-D)

  Indicates the limiting speeds for vehicles in a mounted attack.

- **TRAF.CONTROL** (3-D) (Appendix A)

**Recursive Variable**

**Integer**

- **COUNTER**

  Contains the number of five second move increments that will take place for this entity. This movement update is accomplished by scheduling event QUIK.MOVE.

- **OBS.NUM**

  Receives the value of the field attribute VAR.NLD. This is the sequential id number in order of field creation.
Recursive Variable Real

INTERV

A constant term that is set so that element locations are updated every five seconds during a small portion of the simulation.

Routines Called

HIDER

A routine used to determine the micro-terrain elevation for a selected element.

Temporary Attribute Integer

BLADE_COND

This unit attribute gives the tank the ability to have a dozer blade characteristic.

0 - no blade available
1 - blade available but not in use
2 - blade being used or vehicle is self breaching

DEPMIN

The current position or activity of an element.

1 full defilade
3 turret defilade
4 firing defilade
5 half vehicle defilade
6 moving (defilade determined by the terrain model)
6 reached final area in movement

FLD.NO

Name of the field involved in any pending internal action.

SKILL

Indicates whether an entity has sustained a catastrophic kill.
MKILL

Indicates whether an entity has sustained a mobility kill.

0 no
1 yes

NAM.FLD

This is the sequential id number in order of field creation.

NUM.POINTER

This variable is an argument for event QUIK.MOVE. It carries the number of the entity to be moved.

RAMBER

This variable is an argument for event WALL.BREACH. It carries the number of the entity that made the self breach.

REP.NUMBER

This variable is an argument for event QUIK.MOVE. It carries the number of iterations of scheduling left for this event to perform.

ROUTE

Indicates the number of the route along which the element is travelling.

0 not using a route

SYS.TYPE

This represents the general class of the system of the entity.

1 tanks

326
2 mounted infantry
3 dismounted infantry
4 artillery
5 air
6 air defense
7 bunker
8 comm/ew/acq/intel
9 other

TEAR.DOWN

This variable is an argument for event WALL.BREACH. It carries the number of the field in question.

WPN.TYPE

Describes the specific system within the system code.

Temporary Attribute Real

FLD.INT.DIST

This attribute of an entity contains the distance to a pending field internal action.

P6.FLD

Knowledge level of this obstacle.
0 no knowledge exists
1 knowledge exists and a lane exists

INT.TIME

This variable is an argument for event QUICK.MOVE. It carries the time interval number when the next event is to be performed.

SABIN.FLD

This field attribute is the semi-minor axis length of the elliptical field in meters.
Brief Explanation

Lines 6-8 Prevents unplanned entities from affecting the situation.

Lines 10-12 Determines the appropriate number of five second movement updates to perform to exit this obstacle.

Lines 13-17 Changes the knowledge level to three, a breach exists on this route. It takes the self breacher out of the ditch and lets him shoot again.

CODE

1 EVENT WALL.BREACH GIVEN B,FLD
2 DEFINE B,FLD,OBS.NUM,COUNTER AS INTEGER VARIABLES
3 DEFINE INTERV AS A REAL VARIABLE
4 LET OBS.NUM = WAH.FLD(FLD)
5 LET INTERV = 5.0 "MOVE UPDATES"
6 IF FLD.NC (B) ME FLD RETURN
7 OTHERWISE
8 IF KILL (B) ME 1 AND KILL (B) ME 1
9 LET COUNTER = INT (S.S. SPDS(SYS.TYPE(B),NEW TYPE(B),4))/INTERV
10 SCHEDULE A QUICK.MOVE(B,COUNTER,INTERV) "NOW"
11 LET P6.FLD (FLD) = 3.0
12 LET DEPNUM (B) = 5 "MOVING" CALL HIDER (B)
13 LET BLADE.COND (B) = 0
14 LET TRAP.CONTROL(OBS.NUM,ROUTE(B),3) = RINF.C
15 LET FLD.INT.DIST (B) = RINF.C
16 ALWAYS
17 RETURN
18 END

328
25. **Event GAP.BREACH**

**Purpose**

Event GAP.BREACH simulates the successful breaching of a gap type obstacle by either a vehicle launched bridge or a tank mounted dozer blade. This event is scheduled from routine GAP.INTERNAL.

**Given Arguments**

- Integer
  - B - The Pointer to the entity that has breached the gap obstacle.
  - FLD - Pointer to the field in question.

**Events Scheduled**

- QUICK.MOVE (this appendix)
- STAND.TO (this appendix)

**Global Variable**

- Real
  - LIN.SPDS (3-D)
  - Indicates the limiting speeds for vehicles in a mounted attack.

**Recursive Variable**

- Integer
  - COUNTER
  - Contains the number of five second move increments that will take place for this entity. This movement update is accomplished by scheduling event QUICK.MOVE.
OBS,NUM

Receives the value of the field attribute NAM.FLD. This is the sequential id number in order of field creation.

TMK

Pointer to an element in the platoon of the activating entity.

Recursive Variable Real

INTERV

A constant term that is set so that element locations are updated every five seconds during a small portion of the simulation.

Routines Called

HIDER

A routine used to determine the micro-terrain elevation for a selected element.

Set

PLT.UNIT (1-D)

This set is owned by the permanent entity PLATOON LEADER. It contains the list of temporary entities owned by this platoon.

Temporary Attribute Integer

BAGGED.BOY

This variable is an argument for event STAND.TO. It carries the number of the entity to be moved.
BLADE_COND

This unit attribute gives the tank the ability to have a dozer blade characteristic.

- 0-no blade available
- 1-blade available but not in use
- 2-blade being used or vehicle is self breaching

BREACHER

This variable is an argument for event GAP.BREACH. It carries the number of the entity that made the breach.

DEPNUM

The current position or activity of an element.

- 1 full defilade
- 2 turret defilade
- 3 firing defilade
- 4 half vehicle defilade
- 5 moving (defilade determined by the terrain model)
- 6 reached final area in movement

FLD.FORM

This unit attribute gives the entity the ability to change formations when encountering an obstacle.

FLD.NO

Name of the field involved in any pending internal action.

HOLE

This variable is an argument for event GAP.BREACH. It carries the number of the field in question.

KKILL

Indicates whether an entity has sustained a catastrophic kill.
MKILL

Indicates whether an entity has sustained a mobility kill.

0  no
1  yes

MV STATE

The primary control variable for initiating and stopping movement.

0  In position, do not move.
1  First call to move, do a route select and start to move.
2  Continue movement along a previously selected route.
3  Stop along the route.
4  Next position has been reached, so stop.
5  Final position has been reached, never move again.

NUM.FLD

This is the sequential id number in order of field creation.

NUM.POINTER

This variable is an argument for event QUICK.MOVE. It carries the number of the entity to be moved.

PLT

The number of the platoon to which the entity belongs.

REP.NUMBER

This variable is an argument for event QUICK.MOVE. It carries the number of iterations of scheduling left for this event to perform.
ROUTE

Indicates the number of the route along which the element is travelling.
0 not using a route

SYS.TYPE

This represents the general class of the system of the entity.
1 tanks
2 mounted infantry
3 dismounted infantry
4 artillery
5 air
6 air defense
7 bunker
8 coms/ev/acq/intel
9 other

WPN.TYPE

Describes the specific system within the system code. For example, system type 1 is a tank and weapon type four for this system indicates an AVLB.

Temporary Attribute Real

FLD.INT.DIST

This attribute of an entity contains the distance to a pending field internal action.

FSPEED.FAC

This unit attribute gives the entity the ability to slow down because of obstacle encounter.

INT.TIME

This variable is an argument for event QUIK.MOVE. It carries the time interval number when the next event is to be performed.
P6.FLD

Knowledge level of this obstacle.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no knowledge exists</td>
</tr>
<tr>
<td>1</td>
<td>knowledge exists</td>
</tr>
<tr>
<td>2</td>
<td>knowledge of a lane</td>
</tr>
<tr>
<td>3</td>
<td>existence</td>
</tr>
</tbody>
</table>

P7.FLD

This minefield attribute contains the lane formation number.

P8.FLD

Lane speed factor.

SAVIN.FLD

This field attribute is the semi-minor axis length of the elliptical field in meters.

Brief Explanation

Lines 6-8  Does nothing if the entity has started to move again or if he has left this obstacle. These actions could have been produced as a result of event HEAVY.JUNK actions.

Lines 10-13 Takes the bridge launcher out of the simulation when its bridge is laid over the gap.

Lines 15-17 If the blade tank that scheduled this event has moved because of event
HEAVY-JUNK actions, then this event does not effect his present status.

Lines 19-33 Starts the platoon moving again and puts them in a column formation in order to cross through/over the gap obstacle. The blade tanks that were breaching are slowed down to allow the rest of their platoon to catch up. Enough five second move increments are scheduled to upgrade the detail of the crossing and exit of this field.

Lines 34-35 The knowledge level of this obstacle is changed so that follow-on units can know that a breach exists on this route.

Line 36 The flag indicating an ongoing breaching operation is turned off.

CODE

1 EVENT GAP.BREACH GIVEN B, FLD
2 DEFINE B,FLD, THE, OBS, HUH, COUNTER AS: LENGTHEN VARIABLES
3 DEFINE INTERV AS A REAL VARIABLE
4 LET OBS, HUH = HUH.FLD (FLD)
5 LET INTERV = 5.0 "MOVE UPDATES"
6 IF NV,STATE (B) NE 3 OR FLD.NO (B) WE FLD RETURN
7 OTHERWISE
8 IF SKILL (B) WE 1 AND SKILL (B) WE 1
IF SYS.TYPE(B) EQ 1 AND
(WPN.TYPE(B) EQ 4 OR WPN.TYPE(B) EQ 5)  
LET MV.STATE(B) = 5 "LAUNCHED BRIDGE OUT"
LET DEFNUM(B) = 1 "FULL" CALL HIDER(B)
SCHEDULE A QUIK.MOVE(B,0,0,0) NOW "TO MOVE"
ELSE
IF BLADE.COND(B) ME 2
RETURN
 OTHERWISE
A L W A Y S
FOR EVERY TNK IN PLT.UNIT(PLT(B)) WITH
SYS.TYPE(TNK) ME 3, DO
IF SKILL(TNK) ME 1 AND SKILL(TNK) ME 1 AND
MV.STATE(TNK) ME 5
SCHEDULE A STAND.TO GIVEN TNK NOW
LET DEFNUM(TNK) = 5 "GO" CALL HIDER(TNK)
LET FLD.INT.DIST(TNK) = SAMIN.FLD(FLD)-50.0
LET FLD.FORM(TNK) = INT.F(F7.FLD(FLD))
IF BLADE.COND(TNK) GT 0
LET P8.FLD(FLD)+1)/4
LET BLADE.COND(TNK) =
A L W A Y S
LET COUNTER = INT.F((SAMIN.FLD(FLD)+2.0)/
(.5*LIN.SPDS(SYS.TYPE(TNK),WPN.TYPE(TNK))/INTERV))
SCHEDULE A QUIK.MOVE(TNK,COUNTER,INTERV) NOW
A L W A Y S
LOOP
LET P6.FLD(FLD) = 3.0
LET TRAP.CONTROL(OBS.NUM,ROUTE(B),3) = RINF.C
LET TRAP.CONTROL(OBS.NUM,ROUTE(B),4) = 0.0
A L W A Y S
RETURN
E N D
26. **Event HEAVY.JUNK**

**Purpose**

Event HEAVY.JUNK gives a situation report on the gap breaching operation in progress. This event also initiates the appropriate actions depending on the status update. Event HEAVY.JUNK is scheduled from routines GAP.ENTRY, GAP.INTERNAL, and event HEAVY.JUNK.

**Given Arguments**

*Integer*

\[ B \] - The pointer to the entity that has activated this status check of breaching equipment.

**Events Scheduled**

- **QUIR.MOVE** (this appendix)
- **STAND.TO** (this appendix)
- **TURN.AROUND** (this appendix)

**Global Variable**

*Real*

- **TRAP.CONTROL** (3-D) (Appendix A)
- **LIM.SPDS** (3-D)

Indicates the limiting speeds for vehicles in a mounted attack.

**Recursive Variable**

*Integer*
BRIDGE.STAT
Contains the status of the AVLBs in the platoon.

COUNTER
Contains the number of five second move increments that will take place for this entity. This movement update is accomplished by scheduling event QUIK.MOVE.

EQUIP.STAT
Contains the status of the blade tanks in the platoon.

FLD- Pointer to the field in question.

KNOW.LEVEL
Contains the value of the field attribute P6.FLD. This is the state of intelligence known about the obstacle.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no knowledge exists</td>
</tr>
<tr>
<td>2</td>
<td>knowledge exists</td>
</tr>
<tr>
<td>3</td>
<td>knowledge exists and a lane exists</td>
</tr>
</tbody>
</table>

OBS.NUM
Receives the value of the field attribute NAM.FLD. This is the sequential id number in order of field creation.

TNK
Pointer to an element in the platoon of the activating entity.

Recursive Variable Real
INTERV
A constant term that is set so that element
locations are updated every five seconds during a small portion of the simulation.

SIT.REP.TIME

This is the time between status checks on the breaching operation.

Routines Called

HIDER

A routine used to determine the micro-terrain elevation for a selected element.

JUNK.ALIVE (this appendix)

Set

PLT.UNIT (1-D)

This set is owned by the permanent entity PLATOON LEADER. It contains the list of temporary entities owned by this platoon.

Temporary Attribute Integer

BAGGED.Boy

This variable is an argument for event STAND.TO. It carries the number of the entity to be moved.

BLADE.COND

This unit attribute gives the tank the ability to have a dozer blade characteristic.

0-no blade available
1-blade available but not in use
2-blade being used or vehicle is self breaching

DEFSNUM

The current position or activity of an element.
full defilade
turret defilade
firing defilade
half vehicle defilade
moving (defilade determined by the terrain model)
reached final area in movement

**PLD.FORM**

This unit attribute gives the entity the ability to change formations when encountering an obstacle.

**PLD.NO**

Name of the field involved in any pending internal action.

**KKILL**

Indicates whether an entity has sustained a catastrophic kill.

0 no
1 yes

**HKILL**

Indicates whether an entity has sustained a mobility kill.

0 no
1 yes

**MV.STATE**

The primary control variable for initiating and stopping movement.

0 In position, do not move.
1 First call to move, do a route select and start to move.
2 Continue movement along a previously selected route.
3 Stop along the route.
4 Next position has been reached, no stop.
5 Final position has been reached, never move again.
NAM.FLD
This is the sequential id number in order of field creation.

NUM.POINTER
This variable is an argument for event QUIK.MOVE. It carries the number of the entity to be moved.

PLT- The number of the platoon to which the entity belongs.

REP.NUMBER
This variable is an argument for event QUIK.MOVE. It carries the number of iterations of scheduling left for this event to perform.

ROUTE
Indicates the number of the route along which the element is travelling.

  0  not using a route

STATUS
This variable is an argument for event HEAVY.JUNK. It carries the number of the entity to be updated.

SYS.TYPE
This represents the general class of the system of the entity.

1  tanks
2  mounted infantry
3  dismounted infantry
4  artillery
5  air
6  air defense
7  bunker
8  com/ew/acq/intel
9  other
TWIST

This variable is an argument for event TURN.AROUND. It carries the number of the entity to be turned in the opposite direction.

TYP.FLD

Field type code.
1 - a mandatory dismount field
2 - a minefield
3 - a minefield decision field
4 - a tank ditch
5 - a road crater
6 - a blown bridge (short-wet gap)
7 - a gap decision field

WPN.TYPE

Describes the specific system within the system code.

Temporary Attribute Real

FLD.INT.DIST

This attribute of an entity contains the distance to a pending field internal action.

FSPEED.FAC

This unit attribute gives the entity the ability to slow down because of obstacle encounter.

INT.TIME

This variable is an argument for event QUIK.MOVE. It carries the time interval number when the next event is to occur.

P6.FLD

Knowledge level of this obstacle.
0 no knowledge exists
2 knowledge exists
knowledge of a lane exists

P7.FLD
This minefield attribute contains the lane formation number.

P8.FLD
Lane speed factor.

SAMIN.FLD
This field attribute is the semi-minor axis length of the elliptical field in meters.

T.SPD
The simulation time at which the most recent movement update ended. (the time that SPD was last set.)

Brief Explanation

Lines 11-34 When a breach has been successfully performed, all elements on this cleared route continue to move or start moving again, depending on their current move status. These elements are put into column formations and the breaching equipment is slowed down to allow the rest of the platoon time to catch up.

Line 35 Checks on the current status of the
platoon vehicles performing the breach.

**Lines 36-62** Simulates the actions that happen when the breaching equipment at the gap is depleted from direct/indirect fire.

**Lines 38-45** If a bypass does not exist, the platoon's remaining vehicles start moving to the gap in order to attempt a self breach. The flag for an ongoing breach is turned off.

**Lines 46-59** When the bypass option is still a viable one and the breaching operation has failed, then the platoon is turned around and started back toward the decision ellipse.

**Line 63** If the operation is still being performed by alive platoon equipment, then another breaching operation situation report is scheduled for the future.
CODE

1 EVENT HEAVY,JUNK GIVEN B
2 DEFINE B,FLD,TNK,KNOV,LEVEL,OBS.NUM,EQUIP,STAT,
3 BRIDGE,STAT,COUNTER AS INTEGER VARIABLES
4 DEFINE ST,R EP,TIME,INTERV AS REAL VARIABLES
5 LET FLD = FLD.WO (B)
6 LET KNOV,LEVEL = INT.P(FLD(FLD))
7 LET INTERV = 5.0 "TIME TO UPDATE STATUS
8 LET ST,R EP,TIME = 15.0 "TIME TO UPDATE STATUS
9 LET OBS.NUM = NAM,FLD(FLD)
10 IF KILL (B) NE 1 AND KILL (B) NE 1 AND
11 MV,STAT(E) NE 5
12 IF KNOV,LEVEL EQ 3
13 IF TYP,FLD(FLD) NE 4 OR (TYP,FLD) EQ 4 AND
14 TRAF,CONTROL(OBS.NUM,ROUTE(B),3) EQ RMP,C
15 IF HV,STAT(E) NE 2
16 LET T,SPD (B) = TIME.V
17 LET COUNTER = INT.P((SAWMIN,FLD(FLD)*2.0)/
18 (1.5*LIM,SPD(SYS,TYP(B),WPN,TYP(B),4))/INTERV)
19 SCHEDULE A QUIK,MOVE(B,COUNTER,INTERV) NOW
20 ALWAYS
21 LET HV,STAT(E) = 2 "MOVING
22 LET DEEP,NUR(B) = 5 'MOVING' CALL HIDER(B)
23 LET FLD,INT.DIST(B) = SAMIN,FLD(FLD) - 50.0
24 LET FLD,FORM(B) = INT.P(FLD(FLD))
25 IF BLADE,COND (B) EQ 2
26 LET FSPEED,PAC (B) = (P8,FLD(FLD) +1)/4
27 LET BLADE,COND (B) =
28 ALWAYS
29 IF SYS,TYP (B) EQ 1 AND
30 (WPN,TYP(B) EQ 4 OR WPN,TYP (B) EQ 5)
31 LET FSPEED,PAC (B) = (P8,FLD(FLD) +1)/4
32 ALWAYS
33 LET TRAF,CONTROL(OBS.NUM,ROUTE (B),4) = 0.0
34 OTHERWISE
35 ALWAYS
36 CALL JUNK,ALIVE GIVEN B YIELDING
37 EQUIP,STAT,BRIDGE,STAT
38 IF BRIDGE,STAT NE 1 "NO BRIDGING EQUIPMENT
39 IF TYP,FLD(FLD) EQ 6 OR EQUIP,STAT NE 1
40 "NO BYPASS
41 SCHEDULE A STAND,TO GIVEN B NOW
42 LET DEEP,NUR(B) = 5 'MOVING' CALL HIDER(B)
43 LET FLD,INT.DIST(B) = SAMIN,FLD(FLD) - 50.0
44 LET TRAF,CONTROL(OBS.NUM,ROUTE (B),4) = 0
45 LET COUNTER = INT.P(SAMIN,FLD(FLD) /
46 (1.5*LIM,SPD(SYS,TYP(B),WPN,TYP(B),4))/INTERV)
47 SCHEDULE A QUIK,MOVE(B,COUNTER,INTERV) NOW
48 ELSE "A BYPASS EXISTS/B.B MUST HAVE ONE
49 LET COUNTER = 10 "ENOUGH GET TURNED
50 SCHEDULE A TURN,AROUND GIVEN B NOW
51 SCHEDULE A QUIK,MOVE(B,COUNTER,INTERV) NOW
52 SCHEDULE A STAND,TO GIVEN B IN 3 UNITS
53 FOR EVERY TK IN PLT,UNIT(FLD(B))
54 WITH SYS,TYP (TK) NE 3, DO
55 IF FLD.WO (TK) NE FLD
56 SCHEDULE A TURN,AROUND GIVEN TK NOW
57 SCHEDULE A QUIK,MOVE(TKN,COUNTER,INTERV) NOW
58
ALWAYS
LOOP
LET P.D.INT.DIST(B) = RINF.C

ALWAYS
RETURN

OTHERWISE
SCHEDULE A HEAVY.JUNK GIVEN B IN SIT.REP.TIME UNITS

ALWAYS
RETURN
END
Event GAP.JOCK

Purpose

Event GAP.JOCK gives entities that do not have a bypass, the ability to cover by direct fire a breach already in progress on their route of movement. This event is scheduled from routine GAP.ENTRY and event GAP.JOCK.

Given Argument Integer

B - The pointer to the entity that is trying to support another platoon's breach.

Events Scheduled

STAND.TO  (this appendix)
QUICK.MOVE  (this appendix)

Global Variable Real

TRAP.CONTROL  (3-D)  (Appendix A)
LINSPDS  (3-D)

Indicates the limiting speeds for vehicles in a mounted attack.

Permanent Attribute Integer

F.PLT.UNIT

The first member of the set owned by the permanent entity, PLATOON LEADER.

Recursive Variable Integer

347
COUNTER

Contains the number of five second move increments that will take place for this entity. This movement update is accomplished by scheduling event QUIK.MOVE.

BRIDGE.STAT

Contains the status of the AVLBs in the platoon.

EQUIP.STAT

Contains the status of the blade tanks in the platoon.

FLD- Pointer to the field in question.

OBS.NUM

Receives the value of the field attribute NA.FLD. This is the sequential id number in order of field creation.

TNK

Pointer to the first element in the platoon of the activating entity.

Recursive Variable Real

INTERV

A constant term that is set so that element locations are updated every five seconds during a small portion of the simulation.

SIT.REP.TIME

This is the time between status checks on the breaching operation.

Routines Called

348
A Routine used to determine the micro-terrain elevation for a selected element.

JUNK_ALIVE (this appendix)

Temporary Attribute  Integer

BAGGED.Boy

This variable is an argument for event STAND.TO. It carries the number of the entity to be moved.

DEFNUM

The current position or activity of an element.

1  full defilade
2  turret defilade
3  firing defilade
4  half vehicle defilade
5  moving (defilade determined by the terrain model)
6  reached final area in movement

FLD.NO

Name of the field involved in any pending internal action.

KKILL

Indicates whether an entity has sustained a catastrophic kill.

0  no
1  yes

KKILL

Indicates whether an entity has sustained a mobility kill.

0  no
1  yes
NAM.FLD

This is the sequential id number in order of field creation.

NUM.POINTER

This variable is an argument for event QUICK.MOVE. It carries the number of the entity to be moved.

REP.NUMBER

This variable is an argument for event QUICK.MOVE. It carries the number of iterations of scheduling left for this event to perform.

ROUTE

Indicates the number of the route along which the element is travelling.

0    not using a route

SABOT.SHOOTER

This variable is an argument for event GAP.JOCK. It carries the number of the activating entity.

SYS.TYPE

This represents the general class of the system of the entity.

1 tanks
2 mounted infantry
3 dismounted infantry
4 artillery
5 air
6 air defense
7 bunker
8 com/ew/acq/intel
9 other

UPN.TYPE

Describes the specific system within the system code.
Temporary Attribute Real

FLD.INT.DIST

This attribute of an entity contains the distance to a pending field internal action.

INT.TIME

This variable is an argument for event QUIK.MOVE. It carries the time interval number when the next event is to be performed.

SAIN.FLD

This field attribute is the semi-minor axis length of the elliptical field in meters.

Brief Explanation

Lines 9-16 When a breaching operation is ongoing by another platoon on this route, then a status check is made on this operation. If it is still in progress, another similar status update is scheduled to occur in the future.

Lines 17-22 The flag that indicates an in-progress breaching operation can be lifted/nullified by a successful breach or by a fatal attempt. In any case, the entity starts to move toward the gap obstacle. Routine
GAP.INTERNAL sorts out the appropriate actions to occur next.

**CODE**

1. EVENT GAP.JOCK GIVEN B
2. DEFINE B.OBS.NUM, FLD, TNK, EQUIP.STAT, BRIDGE.STAT,
   COUNTER AS INTEGER VARIABLES
3. DEFINE SIT.REP.TIME, INTERV AS REAL VARIABLES
4. LET FLD = FLD.NO(B)
5. LET OBS.NUM = MAX.FLD(FLD)
6. LET INTERV = 5.0 "TIME TO UPDATE STATUS"
7. IF SKILL(B) NE 1 AND SKILL(B) NE 1
8. IF TRAF.CONTROL(OBS.NUM, ROUTE(B), 4) NE 0.0
9. LET TNK = 
   F.PLT.UNIT(TRAF.CONTROL(OBS.NUM, ROUTE(B), 4))
10. CALL JUNK.ALIVE GIVEN TNK YIELDING
    EQUIP.STAT, BRIDGE.STAT
11. IF EQUIP.STAT = 1 OR BRIDGE.STAT = 1
12. SCHEDULE A GAP.JOCK GIVEN B IN SIT.REP.TIME UNITS
13. RETURN
14. OTHERWISE
15. ALWAYS
16. LET COUNTER = INT.F((SAIN.FLD(FLD)*2.0)/
   (.5*LIN.SPDS(SYS.TYPE(B), WPN.TYPE(B))/INTERV)
17. SCHEDULE A QUIK.MOVE(B, COUNTER, INTERV) NOW
18. SCHEDULE A STAND.TO GIVEN B NOW
19. LET DEFNUM(B) = 5 "MOVING" CALL HIDER(B)
20. ALWAYS
21. LET FLD.INT.DIST(B) = SAIN.FLD(FLD) - 50.0
22. RETURN
23. END
APPENDIX D

USER INPUT INSTRUCTIONS FOR THE STAR ENGINEER MODULE

A. INTRODUCTION

In order to emplace obstacles in a STAR Combat Simulation the obstacles must be planned in advance. Once the user has planned the use of obstacles and their effects on the tactical play of a simulation, then the input values to implement the plans in STAR may be determined. This Appendix consists of instructions for the user to input the required field parameters in order to represent obstacles in STAR. In addition, the necessary parameters for the representation of certain breaching equipment and different tactical options for the commander are also provided. In order to make this very clear, sample input values of the required data for the Engineer Effects Module are also included.

The first step is to determine the X and Y coordinates of the center of each obstacle. Once these values are known, then the size of the obstacle can be approximated using an ellipse. This is possible because an ellipse has a major and a minor axis. The routes of movement of entities
in the simulation may pass through some of the obstacle ellipses. Once the obstacles and routes have been planned, the user is ready to implement these plans in a STAR simulation.

B. OBSTACLE/FIELD REPRESENTATION

The first step in simulating obstacles in STAR is to input the OB.NUM (the number of obstacles in the scenario) and the RT.OB.NUM (the number of routes affected by obstacles). These input parameters are discussed in detail in Appendix A. The next step in creating obstacles on the STAR battlefield entails using the following three outlines of obstacle field attributes. These inputs are placed right after the OB.NUM and RT.OB.NUM numbers.

1. Minefields

The following define the attributes of the field used to simulate a minefield in STAR:

\[ \text{XC.FLD} = X \text{ coordinate of ellipse center.} \]
\[ \text{YC.FLD} = Y \text{ coordinate of ellipse center.} \]
\[ \text{SAJ.Fld} = \text{Length of the semi-major axis of ellipse in meters.} \]
\[ \text{SMN.Fld} = \text{Length of the semi-minor axis of ellipse in meters.} \]
\[ \text{ANGLE.FLD} = \text{Angle of semi-major axis measured} \]
counterclockwise from the east in degrees.

TYPE.Fld = 2

P1.FLD = 0 (not active) or the number of mines in the scatterable minefield.

P1.FLD = 0 or 1 to signify not activated or activated for a patterned minefield.

P2.FLD = Trip wire length in meters for Scatterable Anti-Personnel mines.

P2.FLD = Belt separation in meters for Patterned Anti-Tank mines.

P3.FLD = mine type (1-11)

1 - M70 Scatterable Anti-Tank mine
2 - M56 Scatterable Anti-Tank mine
3 - M21 Hand-Emplaced Anti-Tank mine
4 - M15 Hand-Emplaced Anti-Tank mine
5 - M19 Hand-Emplaced Anti-Tank mine
6 - M74 Anti-Personnel Blast mine
7 - M25 Anti-Personnel Blast mine
8 - M16 Anti-Personnel Frag mine
9 - ADAM Anti-Personnel Frag mine
10 - M74 Anti-Personnel Frag mine
11 - Claymore Anti-Personnel Frag mine

P4.FLD = Probability of detection and avoidance.

P5.FLD = Probability of pushing a dead plow tank

(1-P5.FLD) = offset probability

P6.FLD = Knowledge level of this minefield

0 - No knowledge exists
2 - Knowledge exists
3 - Knowledge exists and a lane exists (its location is known)

P7.FLD = The lane formation
(This and the offset formations are the last formations to appear in the appropriate part input deck.)

P8.FLD = Lane speed factor
(value between 0-1/small is slower)

P9.FLD = Plow activation delay time in seconds

P10.FLD = Plow speed factor
(value between 0-1/should be < P8.FLD)

P11.FLD = Push speed factor
(value between 0-1/should be < P10.FLD)

P12.FLD = Minefield detection distance

a. Input Example for a Patterned Minefield

This is the input data for a patterned/belted minefield.

53000 101500 500 200 90 2 1 50 3 .1 .5 0 3 .8 15 .6 .4 100

The location of the minefield center is:

X coordinate 53000
Y coordinate 101500

The semi-major axis length is 500 meters.
The semi-minor axis length is 200 meters.
The orientation angle of the minefield measured counterclockwise from the east is 90 degrees.
A minefield is type number 2.
The minefield is active, signified by the number 1.
The spread between mine belts is 50 meters.
This minefield was emplaced using M-21 mines (number 3).
The probability of detecting and avoiding a mine is .10.
(tilt rods may be visually detected)
The probability that a tank will push a dead plow tank is .50.
The attacker does not have intelligence about this minefield. (knowledge level is 0)
The lane formation is the third formation in the formation input list for this example.
When a vehicle is travelling on a lane in a minefield, its speed is reduced by 20%. (input number is .8)
It takes 15 seconds for the plow tank to activate its plow.
When a vehicle is plowing, its speed is reduced by 40%. (input number is .6)
When a vehicle is pushing a dead plow tank, its speed is reduced by 60%. (input number is .4)
The distance travelled in a field before visual identification of the minefield by a vehicle is 100 meters.
b. Input Example for a Scatterable Minefield

This is the input data for a scatterable anti-tank minefield.

53000 101500 500 200 90 2 900 0 1 .4 5 0 3 .8 15 .6 .4 30

The location of the minefield center is:

X coordinate 53000
Y coordinate 101500

The semi-major axis length is 500 meters.
The semi-minor axis length is 200 meters.
The orientation angle of the minefield from east is 90 degrees.

A minefield is type number 2.
The minefield is active and has 900 mines in it.
This type mine (anti-tank) does not have trip wires.
(input number is 0)

This minefield was delivered using B-70 mines. (The input number is 1.)
The probability of detecting and avoiding a mine is .40.
The probability that a tank will push a dead plow tank is .50.
The attacker does not have intelligence about this minefield. (The knowledge level is 0.)
The lane formation is the third formation in the
When a vehicle is travelling on a lane in a minefield, its speed is reduced by 20%. (input number is .8)

It takes 15 seconds for the plow tank to activate its plow.

When a vehicle is plowing, its speed is reduced by 40%.

(the input number is .6)

When a vehicle is pushing a dead plow tank, its speed is reduced by 60%. (input number is .4)

The distance travelled in a field before visual identification of the minefield by a vehicle is 30 meters.

2. **Gap Obstacles**

The following defines the attributes on the field used to simulate a road crater, tank ditch or blown bridge (short-wet gap) in STAR:

- **IC.FLD** = X coordinate of ellipse center.
- **YC.FLD** = Y coordinate of ellipse center.
- **SMAJ.FLD** = Length of the semi-major axis of ellipse in meters.
- **SMIN.FLD** = Length of the semi-minor axis of ellipse in meters.
- **ANGLE.FLD** = Angle of semi-major axis measured
counterclockwise from the east in degrees.

\[ \text{TYPE.FLD} = 4 \text{ (tank ditch); 5 (road crater); 6 (blown bridge).} \]

\[ \text{P1.FLD} = \text{The activation level of the obstacle.} \]

\[ \begin{align*}
0 & \text{ - not activated} \\
1 & \text{ - activated}
\end{align*} \]

\[ \text{P2.FLD} = 0 \]

\[ \text{P3.FLD} = 0 \]

\[ \text{P4.FLD} = 0 \]

\[ \text{P5.FLD} = 0 \]

\[ \text{P6.FLD} = \text{Knowledge level of this obstacle.} \]

\[ \begin{align*}
0 & \text{ - No knowledge exists} \\
2 & \text{ - Knowledge exists} \\
3 & \text{ - Knowledge exists and a lane exists (we know where it is)}
\end{align*} \]

\[ \text{P7.FLD} = \text{The lane formation (This can be just a column formation. There is no choice as in the minefield case.)} \]

\[ \text{P8.FLD} = \text{Lane speed factor. (A value between 0 and 1, smaller is slower.)} \]

\[ \text{P9.FLD} = \text{AVLB activation delay time in seconds.} \]

\[ \text{(Typical values range from 180 to 300 seconds)} \]

\[ \text{P10.FLD} = \text{Tank dozer breach time in seconds.} \]

\[ \text{(Typical values range from 360 to 480 seconds)} \]

\[ \text{P11.FLD} = \text{Tank self breach time in seconds.} \]

\[ \text{(Typical values range from 400 to 600 seconds)} \]

\[ 360 \]
P12.FLD = Probability of a self breach getting stuck 
in the ditch.

a. Input Example for a Tank Ditch

This is the input data for a Tank Ditch.

53000 101500 500 200 90 4 1 0 0 0 0 0 0 3 . 8 180 360 400 . 3

The location of the tank ditch center is:
   X coordinate 53000
   Y coordinate 101500

The semi-major axis length is 500 meters.
The semi-minor axis length is 200 meters.
The orientation angle of the tank ditch from east is 90 degrees.

A tank ditch is type number 4.
The tank ditch is an actual/active obstacle signified by the number 1.
The next four entries are zeros.
The attacker does not have intelligence about this tank ditch. (The knowledge level is 0.)
The lane formation is the third formation in the formation input list.

When a vehicle is traveling on a lane through/over a gap obstacle, its speed is reduced 20%. (Input number is .8).
It takes 180 seconds for the AVLB to bridge the gap in this example.

It takes 360 seconds for the blade tank to breach this gap.

It takes 400 seconds for a self breach to take place.

The probability that the vehicle will be immobilized during a self breach is .30.

3. Decision Ellipses

The following defines the attributes on the field used to simulate the decision ellipses for all types of obstacles in STAR: (WARNING: A blown bridge obstacle must always have a bypass route and a bypass movement control point set. The vehicles cannot "bull through" an obstacle of this type.)

\( \text{XC.FLD} \) = X coordinate of ellipse center.

\( \text{YC.FLD} \) = Y coordinate of ellipse center.

\( \text{SMAJ.FLD} \) = Length of the semi-major axis of ellipse in meters. (This must be large enough so that each element in a platoon hits the decision ellipses.) (CAUTION: Make sure routes are not too close to each other. The user cannot let units on one route hit a decision ellipse on
a different route, except when lateral movement is being performed.)

SAMIN.FLD = Length of the semi-minor axis of ellipse in meters.

ANGLE.FLD = Angle of semi-major axis from east in degrees.

TYPE.FLD = 3 (minefields) and 7 (gaps).

P1.FLD = The field attribute that allows the tactical option to bypass an obstacle if the battlefield situation warrants it.

0 - This route has no bypass available, the entity will move to the obstacle and attempt a self-breach. ("bull through")

route number - This is the bypass route to be used.

P2.FLD = The bypass HCP that is on the bypass route.

(if not applicable enter a 0)

P3.FLD = The obstacle number in front of this ellipse. This allows the decision ellipse to check on the level of knowledge about the existence of the obstacle.

0 - This is a decision ellipse on a route that circumvents the obstacle. The only purpose of this decision ellipse is to get lateral

363
moving entities back into route formations.

obstacle number - This is the sequential input number of the obstacle to be encountered in front of this decision ellipse.

P4.FLD = 0 (set to present route during simulation)
P5.FLD = 0 (set to present MCP during simulation)
P6.FLD = 0 (set to a counter during simulation)
P7.FLD = 0 (not used)
P8.FLD = 0 (not used)
P9.FLD = 0 (not used)
P10.FLD = 0 (not used)
P11.FLD = 0 (not used)
P12.FLD = 0 (not used)

a. Input Example for a Minefield Decision Ellipse

This is the input data for a Minefield Decision Ellipse.

53700 101700 100 100 90 3 14 3 1 0 0 0 0 0 0 0 0

The location of the Decision Ellipse center is:

X coordinate 53000
Y coordinate 101500

The semi-major axis length is 100 meters.
The semi-minor axis length is 100 meters.
The orientation angle of this decision ellipse from the
east is 90 degrees.
A minefield decision ellipse is type number 3.
The bypass route is number 14.
The bypass MCP is 3.
The obstacle in front of this decision ellipse is
field number 1.
The next 9 entries are zeros.

C. ARMORED VEHICLE LAUNCHED BRIDGE REPRESENTATION

The target dimensions for the vehicle launched
bridges were found by finding the dimensions of the bridges
on the US AVLB and the Soviet MTU-20 and putting these
bridge types on the M-1 and T-72 tanks, respectfully.
The next line is the US M-1 AVLB.
1 4 1.5 2.24 .0 3.74 4.0 3.7 8.0 11.15 .0 .60
The next line is the Soviet T-72 bridge.
1 5 1.35 1.92 .0 3.27 3.3 3.38 5.2 11.6 .0 .59
These entries are specified as follows:
1. system type
2. weapon type
3. hull height (meters)
4. turret/bridge height (meters)
5. not applicable
6. entire vehicle height (meters)
7. bridge width (meters)
8. frontal hull width (meters)
9. flank hull width (meters)
10. unlaunched/carried bridge width (meters)
11. not applicable
12. the percent of total vehicle height above
    hull defilade.
The above dimensions are for future systems. If today’s arsenal is to be represented, then actual vehicle configurations and dimensions must be used.

Limiting speeds for these bridges have to be added to the input deck along with the dismounted limiting speeds.

Bridges neither carry nor shoot ammunition. This characteristic can be simulated by putting a zero in the types of ammunition block on input. An example follows:

```
1 4 0
1 5 0
```

The first entry is the system type.
The second entry is the weapon type.
The third entry specifies the different ammunition types for this vehicle.

The array POINT.HOLD should be modified as follows:

```
1 4 9 0 0 0 0 0 0 0 0 0 0 0 0 0
```

The system/weapon target selection array for the vehicle launched bridges have to be added so that they are unable to select targets for firing.

```
1 0 0 0 0 (0-1000 meters)
1 0 0 0 0 (1000-2000 meters)
1 0 0 0 0 (> 2000 meters)
```

The same additions have to be made for the T-72 bridge.

```
1 5 9 1 0 0 0 0 0 0 0 0 0 0 0 0
```
These new entities, Armored Vehicle Launched Bridges, must also be added to the target selection arrays of their respective enemy entities.

The suppression data for each type bridge must be added to the input deck.

D. OFFSET FORMATIONS

The next area of input that must be discussed is the lane and offset formations. These new formations allow the Engineer Module to simulate lanes, breaching operations, and the detour of elements around killed vehicles. The following example is a sample of a FORM OFFSET array.

```
  7  1  5  0  0  0 -50  0  50  0 -100  0  100
  1  5  0  0  0  90  0 -50  0  100  0 -100  0
  1  5  0  0  0  50  0 -100  0 -150  0 -200  0
  4  3  5  0  0  10  -50  10  -100  0 -150  0 -200  0
  4  3  5  0  0  20  -50  10  -100  20 -150  20 -200  20
  4  3  5  0  0  30  -50  30 -100  30 -150  30 -200  30
  7  5  0  40  -50  40 -100  40 -150  40 -200  40
```

The first number (7) indicates that there are seven formations to follow.

The first formation in the list is a five place line formation.

column 1 specifies this formation as number one for the program.
column 2 Specifies that there are five positions in this formation.

column 3-4 Specify the X and Y offsets for the vehicle.

This entity has a formation position equal to one.

Each succeeding pair of numbers specifies the formation offsets for vehicles with formation positions which are in increments of one. A vehicle's formation position is the same as his POS.IN.PLT.AREA.

The second formation is a five place column formation with tank one (0,0), in the middle of the platoon.

The third formation is a five place breaching formation with the first tank in the lead. This tank should be the plow tank for this platoon. Tank one can be the plow tank if the element data is assigned in the proper manner.

The fourth formation is the first offset formation. This formation would be the first used if a dead vehicle became an obstacle to movement on the route. All elements are in the same relative positions within their platoon as dictated by formation number three, the breach formation, except that the platoon has been displaced 10 meters to the left of its route.
The fifth, sixth, seventh formations are the same as formation number three, except offsets of 20, 30, 40 meters, respectively, have been added as Y-displacements off the route. The number of offset formations, not including the lane/breach formation, depends on the number of mechanized entities in the largest platoon in the scenario. The number of offset formations should be at least this large.

E. TANK BREACHING ATTACHMENTS

The Mine Plow and Tank Dozer Blade characteristics for individual entities are submitted for the simulation as part of the element data in the input deck. The input position definitions are defined below.

position:

1 NAME The element number of the entity.
2 SYS.TYPE This represents the general class of the system of the entity.
   1 Tanks
   2 Mounted infantry
   3 Dismounted infantry
   4 Artillery
   5 Air
   6 Air defense
   7 Bunker
   8 Comm/ev/acq/intel
   9 Other
3 WPN.TYPE Describes the specific system within the system code. For example, system type is a tank and weapon type of seven

369
for this system indicates a Soviet T-72 tank.

4 SEC Indicates the number of the SECTION to which the entity belongs.

5 PLT The number of the PLATOON to which the entity belongs.

6 CO The number of the COMPANY to which the entity belongs.

7 BN Indicates the number of the BATTALION to which the entity belongs.

8 SQDVEH Indicates the number of the entity's squad vehicle.

9 PLTLDR The element number of the entity's PLATOON LEADER. The PLATOON LEADER must be the first element input in his PLATOON.

10 COCDR The element number of the entity's COMPANY COMMANDER. The COMPANY COMMANDER is the first entity input in his COMPANY.

11 START AREA The number which identifies the starting movement area for this entity.

12 ALIVE.DEAD Indicates whether the entity is alive.
or dead.

0   Alive
1   Dead
2   Alive mounted in carrier.

13  FIR.MODE Indicates the mode of fire that a
dismounted element is to use.

0   Fires in a semi-automatic mode.
1   Fires in an automatic mode.

14  PLOW.COND This unit attribute gives the tank
    a mine plow.

0   No plow
1   Plow attached

15  BLADE.COND This unit attribute gives the tank
    a bull dozer blade.

0   No blade
1   Dozer blade attached

The following is an example of two typical element data
entries:

45 1 7 45 15 4 2 45 45 40 601 0 0 1 0
46 1 7 46 15 4 2 46 45 40 601 0 0 0 1

In this example - Element number 45 is a Soviet T-72 tank
    with a mine plow attachment.
- Element number 46 is a Soviet T-72 tank
    with a dozer blade attachment.

In this appendix, the authors have not tried to give
complete user input instruction for STAR. The input instructions given are enough for the current user of the STAR model to run the model with the Engineer Effects Module. If more complete instructions are required, then the reader is directed to the STAR Combat Model documentation currently in use.
LIST OF REFERENCES


11. Department of the Army, FM 5-34, Engineer Field Data, September 1976.


**INITIAL DISTRIBUTION LIST**

<table>
<thead>
<tr>
<th>No.</th>
<th>Copies</th>
<th>Addressee</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>2</td>
<td>Defense Technical Information Center</td>
<td>Cameron Station Alexandria, Virginia 22314</td>
</tr>
<tr>
<td>2.</td>
<td>2</td>
<td>Library, Code 0142</td>
<td>Naval Postgraduate School Monterey, California 93940</td>
</tr>
<tr>
<td>3.</td>
<td>1</td>
<td>Department Chairman, Code 55</td>
<td>Naval Postgraduate School Monterey, California 93940</td>
</tr>
<tr>
<td>4.</td>
<td>10</td>
<td>Chief</td>
<td>Naval Postgraduate School Monterey, California 93940</td>
</tr>
<tr>
<td>5.</td>
<td>5</td>
<td>Associate Professor James K. Hartman</td>
<td>Naval Postgraduate School Monterey, California 93940</td>
</tr>
<tr>
<td>6.</td>
<td>5</td>
<td>Associate Professor S. H. Parry</td>
<td>Naval Postgraduate School Monterey, California 93940</td>
</tr>
<tr>
<td>7.</td>
<td>1</td>
<td>Headquarters</td>
<td>US Army Training and Doctrine Command ATAC: Director, Studies and Analysis Directorate, Mr. G. Goldberg Fort Monroe, Virginia 23651</td>
</tr>
<tr>
<td>8.</td>
<td>1</td>
<td>Associate Professor A. L. Schoenstadt</td>
<td>Naval Postgraduate School Monterey, California 93940</td>
</tr>
<tr>
<td>9.</td>
<td>1</td>
<td>Professor James G. Taylor</td>
<td>Naval Postgraduate School Monterey, California 93940</td>
</tr>
</tbody>
</table>

376
<table>
<thead>
<tr>
<th>Number</th>
<th>Address 1</th>
<th>Address 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Office of the Commanding General</td>
<td>ATTN: General Glenn Otis</td>
</tr>
<tr>
<td></td>
<td>US Army Training and Doctrine Command</td>
<td>Fort Monroe, Virginia 23651</td>
</tr>
<tr>
<td>11</td>
<td>Headquarters</td>
<td>ATTN: ATCG-T (BG Morelli)</td>
</tr>
<tr>
<td></td>
<td>US Army Training and Doctrine Command</td>
<td>Fort Monroe, Virginia 23651</td>
</tr>
<tr>
<td>12</td>
<td>Office of the Commanding General</td>
<td>ATTN: General Donn A Starry</td>
</tr>
<tr>
<td></td>
<td>US Army Readiness Command</td>
<td>MacDill AFB, Florida 33621</td>
</tr>
<tr>
<td>13</td>
<td>Deputy Under Secretary of the Army</td>
<td>ATTN: G.E. Creel</td>
</tr>
<tr>
<td></td>
<td>for Operations Research</td>
<td>Room 2E281, Pentagon</td>
</tr>
<tr>
<td></td>
<td>Washington, DC 20310</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>LTG Howard Stone</td>
<td>ATTN: LTG Howard Stone</td>
</tr>
<tr>
<td></td>
<td>Commanding General</td>
<td>Fort Leavenworth, Kansas 66027</td>
</tr>
<tr>
<td>15</td>
<td>Director</td>
<td>ATTN: ATZL-CAC-A (Mr. Lee Pleger)</td>
</tr>
<tr>
<td></td>
<td>Combined Arms Combat Development Activity</td>
<td>Fort Leavenworth, Kansas 66027</td>
</tr>
<tr>
<td>16</td>
<td>Director</td>
<td>ATTN: KM Pickett</td>
</tr>
<tr>
<td></td>
<td>Combat Analysis Office</td>
<td>US Army Combined Arms Center</td>
</tr>
<tr>
<td></td>
<td>ATTN: KM Pickett</td>
<td>Fort Leavenworth, Kansas 66027</td>
</tr>
<tr>
<td>17</td>
<td>Commander and General Staff College</td>
<td>ATTN: Education Advisor</td>
</tr>
<tr>
<td></td>
<td>ATTN: Education Advisor</td>
<td>Room 123, Bell Hall</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fort Leavenworth, Kansas 66027</td>
</tr>
<tr>
<td>18</td>
<td>Dr. Wilbur Payne, Director</td>
<td>ATTN: Dr. Wilbur Payne</td>
</tr>
<tr>
<td></td>
<td>US Army TRADOC Systems Analysis Activity</td>
<td>White Sands Missile Range, New Mexico 88002</td>
</tr>
<tr>
<td>19</td>
<td>Headquarters, Department of the Army</td>
<td>ATTN: BAG-2D</td>
</tr>
<tr>
<td></td>
<td>Office of the Deputy Chief of Staff for Operations and Plans</td>
<td>Washington, D.C. 20310</td>
</tr>
<tr>
<td>20</td>
<td>Commander</td>
<td>ATTN: BAG-2D</td>
</tr>
<tr>
<td></td>
<td>US Army Concepts and Analysis Agency</td>
<td>8120 Woodmont Avenue</td>
</tr>
<tr>
<td></td>
<td>ATTN: BAG-2D</td>
<td>Bethesda, Maryland 20014</td>
</tr>
</tbody>
</table>
21. **Director**
   US Army Material Systems Analysis Activity
   ATTN: DRAST-CN (Mr. Bill Niemeyer)
   Aberdeen Proving Grounds, Maryland 21005

22. **Director**
   US Army Night Vision and Electro-Optical Lab
   ATTN: DEL-NV-VI (Mr. Frank Shields)
   Fort Belvoir, Virginia 22060

23. **Director**
    USATRASAMA
    ATTN: Mr. Ray Heath
    White Sands Missile Range, New Mexico 88002

24. **Director**
    Combat Developments, Studies Division
    ATTN: MAJ W. Scott Wallace
    US Army Armor Agency
    Fort Knox, Kentucky 40121

25. **Commandant**
    US Army Field Artillery School
    ATTN: ATSA-CD-FAWS
    Fort Sill, Oklahoma 73503

26. **Director**
    Combat Developments
    US Army Aviation Agency
    Fort Rucker, Alabama 36362

27. **Director**
    Combat Developments
    US Army Infantry School
    Fort Benning, Georgia 31905

28. **Director**
    Combat Developments
    ATTN: ATSA-CDR (CPT James Hudd)
    US Army Engineer School
    Fort Belvoir, Virginia 22060

29. **Director**
    Combat Developments
    ATTN: ATSA-CDF-S
    US Army Air Defense Agency
    Fort Bliss, Texas 99885

30. **Commander**
    US Army Logistics Center
    ATTN: ATLS-LOS (Br. Cameron/CPT McGrann)
    Fort Lee, Virginia 23801
|   | 31. Commandant  
US Army Signal School  
ATTN: LTC Harnagel  
Fort Gordon, Georgia  
30905 |
|---|---|
|   | 32. CPT James V. Rudd  
35 Neussy Ave.  
Buffalo, New York 14220 |
|   | 33. CPT Stephen C. Main  
US Readiness Command  
ATTN: ASG  
MacDill AFB, Florida 33621 |